TECHNISCHE UNIVERSITÄT
CHEMNITZ

Department of Artificial Intelligence

Inhibition and loss of information in unsupervised feature extraction

Dissertation

submitted for the degree of

Doktor der Ingenieurwissenschaften (Dr.-Ing)

in the

Faculty of Computer Science,
Technische Universität Chemnitz

by Arash Kermani Kolankeh

Chemnitz, March 15, 2018

Examiners: Prof. Dr. Fred H. Hamker, Technische Universität Chemnitz
Prof. Dr. Vladimir Grigorievich Spitsyn, Tomsk Polytechnic University

Kermani Kolankeh, Arash

Email: arash@informatik.tu-chemnitz.de

Dissertation: Inhibition and loss of information in unsupervised feature extraction

Faculty of Computer Science, Technische Universität Chemnitz

Chemnitz, March 15, 2018

Abstract

In this thesis inhibition as a means for competition among neurons in an unsupervised learning system is studied. In the first part of the thesis, the role of inhibition in robustness against loss of information in the form of occlusion in visual data is investigated. In the second part, inhibition as a reason for loss of information in the mathematical models of neural system is addressed. In that part, a learning rule for modeling inhibition with lowered loss of information and also a dis-inhibitory system which induces a winner-take-all mechanism are introduced. The models used in this work are unsupervised feature extractors made of biologically plausible neural networks which simulate the V1 layer of the visual cortex.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Understanding the function of the Visual system as an ideal object recognition system could help us to enhance the existing models and methods for object recognition.

The abstract representation of the data is an essential ability of the visual system. This abstract representation helps object recognition with shift and rotation invariance and brings robustness to different kinds of noise and occlusion.

The natural visual system has started its job by learning the building blocks of the visual world. The most basic of these building blocks or components are learned in the primary visual cortex over the evolution in the form of edge detectors. This first layer of the visual cortex called V1 contains neurons called the "simple cells" which act like Gabor filters (for a detailed explanation of Gabor filter, please refer to the Appendix). Each Gabor filter or respectively each V1 simple cell reacts to an edge with a particular frequency, direction and coordinate (location) in the visual field.

The process of learning in V1 simple cells has been studied and simulated by many authors (Olshausen and Field [1996], Bell and Sejnowski [1997], Hoyer and Hyvärinen [2000], Falconbridge et al. [2006], Rehn and Sommer [2007], Wiltschut and Hamker [2009], Spratling [2010], Zylberberg et al. [2011]). From the engineering point of view, each Gabor filter learned by a V1 simple cell is a feature explaining the visual environment. Features are building blocks of data and the process of exploring these features is called Feature Extraction. Thus, feature extraction is the first step in object recognition. The features in the visual system though, are "learned" or better to say, the feature ex-

traction process is called learning here. Mathematical modeling of the learning process is a key step in building a system which could simulate the function of the visual system. In this thesis, the mathematical models of neurons mainly resemble the V1 simple cells which as mentioned, are edge detectors.

One could look at feature extraction like at an optimization problem in which the goal is finding features, often understood as vectors in the space of data, which could describe the world as well as possible. Extracting features from the visual data could be done in supervised or unsupervised manner. An explanation of supervised and unsupervised methods could be found in the Chapter 2. In a supervised manner, the optimal output of the system is predefined, and the features are learned while the system approaches its optimal output. On the other hand, in unsupervised methods, the system lacks a global criterion for its optimality. However, the type of the feature extraction under attention in this work is the unsupervised feature extraction like what has naturally happened in V1.

Because of the similar nature of the V1 simple cells and because these neurons are not guided by a global signal, they could theoretically learn very similar features. This could disable the visual system from understanding the visual world using this almost unique feature learned by different neurons. Here, the concept of competition as a means of achieving non-similar features comes to importance.

Competition as a means to extract differing features in an unsupervised manner: The interactions among neurons could be divided to excitatory and inhibitory, meaning that some neurons stimulate the others and some prevent the others from firing. Occupying rather a small portion (15% to 20%) of the cells in the brain, inhibitory cells play essential roles in the functionality of the brain. Of their functionalities one could count: the role of inhibitory neurons in ending the critical period of learning after childhood Kuhlman et al. [2013]; stabilization; the way the brain produces sparse codes as a result of inhibition Rozell et al. [2008], Gregor et al. [2011], Xiao and Liu [2015], Stein and Brito [2015], Zylberberg et al. [2011]; role of inhibition in edge enhancement in the visual cortex Fernandes et al. [2013], Arkachar and Wagh [2007], etc. Inhibition induces competition by preventing co-activation (simultaneous activation) of the neurons.

From one point of view, inhibition causes competition by favoring some units of the

brain to the other ones or better to say, by suppressing some of them. In a model of the nervous system, a neuron with suppressed activity is prevented from learning. If under the presence of a particular input one neuron is stimulated by the input but not suppressed by the other neurons, it will have a high activity and as a result, its synaptic weights will be increased as a function of the correlation between the input and the output (pre- and post-synaptic) values. On the other hand, the neuron which is suppressed will not be capable of learning from the current input. Of course, in most of the cases, neurons are not completely suppressed, but their strength of learning from the input is reduced relatively by the strength of the inhibition they receive. This way, the neurons learn from the current input with different speeds and as a result, they learn different features from the whole input space.

In Kermani Kolankeh et al. [2015] which has made the section 3 of the thesis, the behavior of three different biologically plausible models which are capable of learning Gabor filters from natural images are compared. It was observed that models which more intuitively suppress the unnecessary co-activations of their units are more robust to distortions of the input in the form of occlusion.

A serious problem which might occur in a model with inhibition is the loss of information. Incredibly strong inhibition could prevent neurons from learning and also from passing their activities to the higher processing layers. This problem is discussed in section 4.1.3 and a solution is provided. The solution consists of a dynamic learning rule for the lateral inhibitory weights which imply competition to the system. The introduced solution regulates the inhibitory weights in a way to prevent them from suppressing the neurons when suppression causes loss of information.

In popular models of neural feature extraction like in convolutional neural networks (CNN) LeCun et al. [1998] robustness is achieved not through competition but using pooling. In this work, a comparable mechanism with pooling which is also based on learning from aggregation of activities is used. This mechanism, called dis-inhibitory (DI) which is based on the function of Vasoactive Intestinal Polypeptide (VIP) cells in the brain Rudy et al. [2011] is introduced in chapter 4. The dis-Inhibition mechanism was invented to bring more invariance to the cells of the second layer of the two-layer network proposed here. Dis-inhibition induces aggregation of information by dis-inhibiting the winner in a neighborhood and further inhibiting its neighbors. This

lets the winner learn based on the aggregation of information its neighbors would have learned from.

The goal of this thesis is a better understanding of the inhibition and competition in the neural system and their relation to the loss of information and robustness. Section 3 talks about the robustness against loss of information achieved through competition and section 4 talks about the loss of information caused by the inhibitory mechanism itself and the ways one could prevent it.

Organization of thesis: This thesis starts with an introduction to different learning methods in Chapter 2. Chapter 2 ends with an explanation about non-generative unsupervised learning methods to which also the Hebbian based learning methods used in this thesis belong. As the introduced one- and two-layer models try to picture aspects of learning which are or could be utilized in deep learning methods, a part of the thesis is dedicated to deep learning. Especially because dis-inhibition mechanism introduced in this work is comparable to pooling which also has its roots in complex-cells, the section 2.3.1 gives an explanation about pooling methods used in deep learning.

The practical work is divided into two parts. The first part (Chapter 3), also published in Kermani Kolankeh et al. [2015] shows the importance of competition in overcoming loss of information in the form of occlusion in the visual data. Chapter 4 contains the second part of the practical work. Section 4.1.3 focuses on the balance between excitation and inhibition by defining an innovative method for learning inhibitory weights. The criteria for performance were shift, rotation and noise invariance. Section 4.1.3 is about a dis-inhibitory mechanism and its effect on improvement of the invariance mentioned in the second part.

Chapter 2

State-of-the-art

Basically, one could divide learning methods into three main groups: unsupervised, supervised and hybrid (Deng [2013]). Here an introduction to unsupervised and supervised methods is provided. In both of these groups, the goal is making decisions (classification, regression) based on the transformed data.

One difference between supervised and unsupervised methods is that in the supervised methods, the optimization objective of the model is to try to reduce the error in the decisions of the model. On the other hand, in unsupervised methods, the goal is finding a better representation of the data by finding out the essential features which shape the data.

Most of the unsupervised classification or regression systems have two parts; one is a set of feature extractors, and the other one is a classifier or a regressor. Such models are optimized in two different stages; first feature extraction (data transformation) and then, decision-making, that is, classification or regression.

2.1 Supervised learning

As mentioned above, in supervised methods the model is optimized with the goal of reducing the error or the difference between the label (desired output) and the actual output of the model. The fact that we use labeled data gives the models the name "supervised". One of the most successful learning methods which has a high perfor-

mance in classification of images is Convolutional Neural Network (CNN) (Fukushima [1980],LeCun et al. [1998]). CNN is based on one of the first generations of the neural networks, that is Multi-Layer Perceptron (MLP) and is the base of many deep learning methods. In the following section a short introduction to MLP and then an explanation about CNN are provided.

2.1.1 Multilayer Perceptron

The multilayer perceptron is the base of many other neural network methods. It is made of several layers of artificial neurons. Each neuron receives some parallel input signals. Each of these inputs is multiplied by a coefficient called synaptic weight, and the results are summed up to make the "net" response of the neuron. The output of the neuron is a function of this net value. This function is called the activation function. Each neuron is actually an axis in the new space to which the input space is transferred and the net output is the projection of the input point to this axis. What activation function does is, emphasizing the non-linear differences among projections.

Each layer of neural network consists of a number of neurons in a row which receive the same set of inputs. One may think of MLP as a stack of layers of neurons in which, the output of each layer is the input to the higher layer. Imagining this stack like a pyramid, based on the number of the neurons on the top of the pyramid and the activation functions used, one could do a classification or regression task. A pyramid with one neuron on the top does regression and a pyramid with more than one output could be used as a classifier. In both of these cases, the lower layers try to find an abstract representation of the data which could be easily (linearly) separable for the higher layers. It is theoretically proven that any non-linear function could be approximated with MLP if the network is properly designed and has a proper number of hidden units.

Looking at MLP as an optimization problem, the objective is to reduce the difference between the desired output (label) and the actual output. This difference is also called the error. This minimization is usually done using backpropagation of the error from the higher layers to the lower layers and updating the weights based on gradient descent.

2.1.2 Convolutional Neural Network

Learning abstraction through multilayer feature extraction: The sensory system receives lots of noisy data. Among all those noisy information, what makes sense is considered for further processing, and the rest is discarded. We can recognize a car coming to our direction even if it has an unfamiliar appearance which has not been observed before. We can identify a combination of two points and one pretense as a smile (:)). These are examples of high-level abstract information processing in the human brain. This abstraction is achieved by several stages of feature extraction. As for the smile, on the first layers eyes and lips are recognized and on the higher layers combinations of these features which are associated with happiness. In modeling the hierarchy of information processing in the brain also, the same concept is considered. On each layer, the information becomes more abstract, and on the last layer, regression or classification is applied to the information. These methods are usually called Deep Learning methods due to their multi-layer architecture. Although classification of the raw input might need non-linear classifiers, the output of such models could be classified with linear classifiers. In addition, a good abstraction of the data could overcome noise and meaningless variances in the data. Such models are capable of finding patterns independent from the background facts. Convolutional neural networks are of the most popular and documented deep learning methods, which will be discussed in this section.

Convolutional Neural Networks have been particularly designed for object recognition, and their success lies in their robustness to variances. CNNs are based on the same principles as MLP; a pyramid of layers of neurons is optimized with backpropagation of error to minimize the error. However, there are some essential differences between these two methods: CNN is shift-invariant and rotation invariant. These abilities have come from using pooling layers which are absent in MLP. The other main difference is the concept of weight sharing which speeds up the computation and reduces the complexity of learning and classification by reducing the number of parameters. The latter one is important especially because CNN is trained and used in relatively huge architectures. One other advantage of weight sharing is reducing the capacity of the network and making the learning possible on small train sets LeCun et al. [1998]. Pooling layers mentioned, bring shift invariance to the network by passing the maximum or mean of a

neighborhood of activities to the higher layer. When for example a max-pooling layer is located between two layers of neurons, the pooling layer sends the activity of the maximally active neuron in each group of neurons in the lower layer to the higher layer. This way, the neurons of the higher layer will be aware of the appearance of a pattern in a part of the lower layer no matter where it is located in a neighborhood of the lower layer neurons.

Weight sharing is what actually simulates **convolution** in a CNN. Convolution is basically the calculation of the dot product of a surface to a sliding filter on that surface. We imagine to have a surface of copies of the same filter or neuron and update just one weight matrix to minimize the error this surface produces. This surface of similar neurons is called a map which when applied to the input, produces an output surface. On the output surface in the locations where the input surface matches with the filter or neuron stronger activities will be observed. One other reason for using shared weights is that a feature found in one part of the image could also describe other parts of the image. It is obvious that we are not looking for just one feature on the input surface and several maps on each layer are needed.

A famous version of convolutional neural networks is LeNet-5 whose architecture is depicted on Figure 3.1 LeCun et al. [1998].

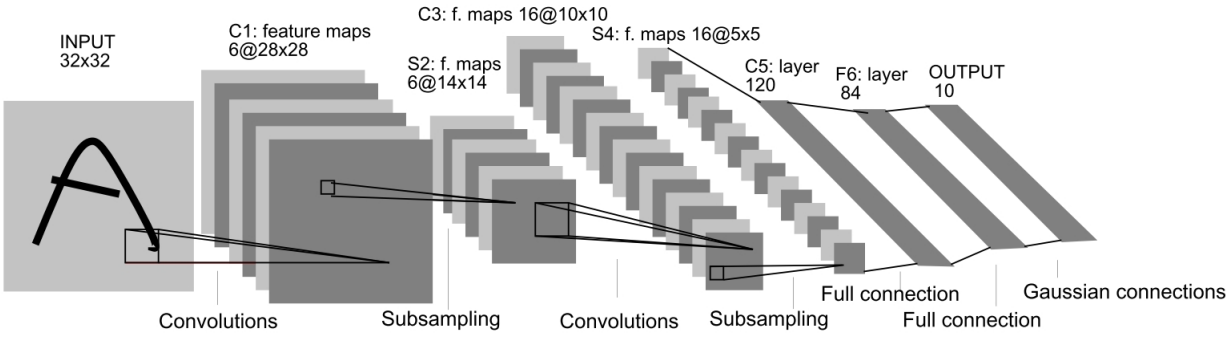


Figure 2.1: LeNet-5, a convolutional neural network with 7 layers. Each C (convolutional) layer is followed by a sub-sampling (mean pooling) layer except for the last CN layer C5, which is followed by an all-to-all connection to the F6 layer, forming a classifier.

2.2 Unsupervised learning

The word "unsupervised" in this context means without guide or teacher. Practically this means learning from unlabeled data. Although most of the popular high-performance methods in the history of deep learning are supervised or at-least hybrid (supervised and unsupervised), unsupervised methods have recently come to importance as the big data, cannot be labeled. Labeling is a task which is done by the human. That is, calling an object by a name. There exist unsupervised works (Dosovitskiy et al. [2014]) which give unlabeled objects some fake (surrogate) labels to improves the discrimination ability of models in the learning phase. However, basically unsupervised methods try to extract the features or building blocks of the data which could better explain the data to a classifier or regressor. Unsupervised methods themselves can be divided into two groups: generative and non-generative. In both of these groups, one tries to ensure that the new representation (projection) of the data is meaningful with the least loss of information and the highest level of abstraction. The difference between these two groups is in the ways they seek these objectives.

2.2.1 Generative unsupervised learning

In generative methods, on each level of information processing, one ensures correctness of transformation by trying to reconstruct the input from the code the units on that level produce. Of the generative methods one could think of Matrix Factorization (Hoyer [2004], Predictive Coding Spratling [2016]), Auto-encoders (Bengio [2009]) and Restricted Boltzmann Machines (Norouzi et al. [2009]).

Independent Component Analysis

In dimensionality reduction as one of the first steps in data analysis, one tries to reduce the number of characteristics or fields of the data one takes into the account for analyzing. This is done in different ways. In some cases, one experimentally concludes that some fields of the data do not contain meaningful information or observe that just a few fields are playing the main role in describing the data space. There exist several methods which could automatically look for the more important factors of the data. Of those, one could count Principle Component Analysis or PCA. The main idea of PCA is

finding directions in the data space in which the variance is maximized. The restriction is that these directions should be orthogonal to each other. The procedure is so that first we look for the direction of the biggest variance of the data called the first principal component (PC), then we look for directions orthogonal to the first PC and chose the one with the highest variance and call it the second PC and so on. At the end, one could discard a number of the last PCs which have the lowest variances and thus are less informative for us. Then one could project the data to the new space with a reduced number of dimensions and then further process the data.

There exist more sophisticated methods than PCA, like Independent Component Analysis (ICA). In ICA, one could look for an enough number of interesting components by trying to find directions which share the least amount of information, that is, are statistically independent. The problem can be stated as

$$x = Vy$$

or

$$y' = Wx$$

where V is the mixing matrix and W its inverse. x is the input vector and y' is an approximation of the vector of sources y, or components, which should be independent. ICA, as a generative method, tries to generate the inputs as a sum of components y weighted by the weights of the mixing matrix V.

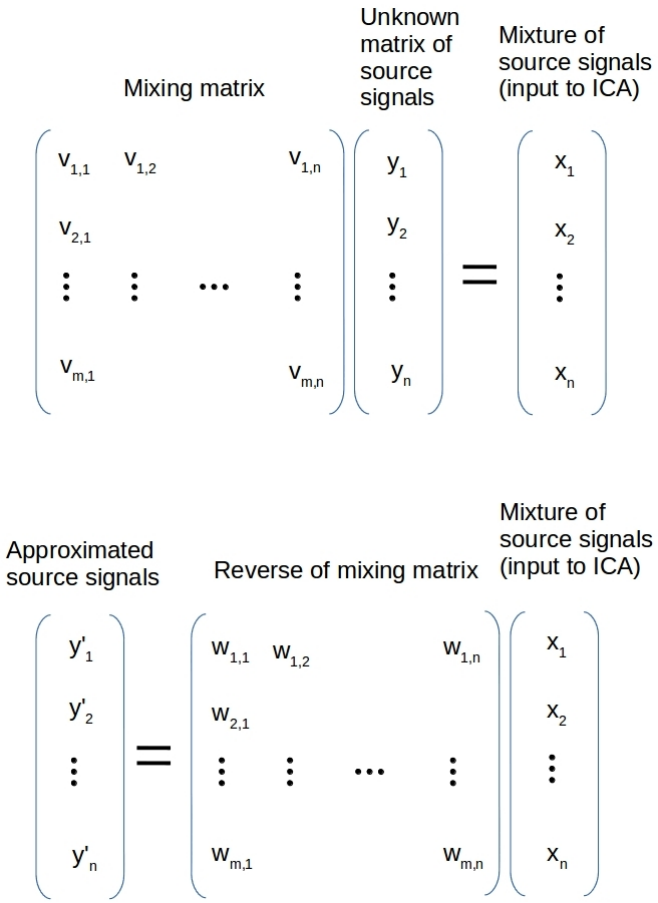


Figure 2.2: Independent components analysis tries to find out the mixing matrix V reverse of which W is used to decode the mixed signal X to its building components Y.

Unlike PCA which is based on the second-order dependencies (variance), ICA tries to find even more complicated dependencies. The goal of Independent Component Analysis is finding a linear transformation for data so that the components are as independent as possible. One could refer to the famous problem of the cocktail party in which different voice sources should be separated based on their statistical independence (Hyvärinen and Oja [2000]). Although ICA has various applications in signal and image processing, from the point of view of computational neuroscience, it could be considered as an approach to simulate some parts of the nervous system like the primary visual cortex (van Hateren and van der Schaaf [1998]).

Fast independent component analysis

As the mixture of several independent variables is always more Gaussian than each of the variables, one way to seek independent components, is to try to decrease gaussianity

as much as possible. In fast independent component analysis (FastICA, Hyvärinen and Oja [1997]), the goal is finding statistically independent components of the data by maximizing neg-entropy. Neg-entropy is a measure of non-gaussianity and non-gaussianity is in direct relation to independence; the more non-Gaussian the activity distributions, the more independent are the components. This could be explained based on the fact that a strongly non-Gaussian distribution has a wide projection on one of the dimensions and very narrow projection on the other dimensions. Tha is, this distribution could be explained as a function of one of the dimensions and independent from the other dimensions. Some other methods for increasing non-guasianity are "minimization of mutual information", "maximum likelihood estimation" or "maximization of joint entropy of nonlinearly transformed neural activities" (Bell and Sejnowski [1997]). Fast independent component analysis is practically used in the current study and is explained in Section 3.2.2.

Non-negative Matrix Factorization

In non-negative matrix factorization with sparseness constraint (NMFSC,Hoyer [2004]), the goal is to factorize the matrix of the input data into non-negative components and non-negative source matrices, imposing more biological plausibility in comparison to FastICA by preventing activities from being negative. NMFSC approaches the matrix of components V to satisfy $X \approx V \otimes Y$. Where Y is the matrix of output vectors and X the matrix of corresponding input vectors. Y and V are calculated while approaching the objective of reducing the difference between the input X and its reconstruction $V \otimes Y$ in a divisive way:

$$V \leftarrow V \otimes (XY^T) \oslash (VYY^T) \tag{2.1}$$

Where $\otimes$ means element-wise multiplication and $\oslash$ element-wise division. One could say that the term $(XY^T) \oslash (VYY^T)$ is actually the modulated input which is used to update V. In some literature, this is interpreted as a divisive form of feedback inhibition (Kompass [2007], Spratling et al. [2009]). This method, introduced by Lee and Seung [1999], tries to minimize the difference between the distributions of the input and its

reconstruction based on Kullback-Leibler divergence.

In some other works, this process is done by adding the subtractive difference between the input and its reconstruction to V. However, the application of both subtractive difference and the divisive modulated input could be understood as the inhibiting input (Spratling et al. [2009]).

The advantage of NMFSC to pure non-negative matrix factorization (Lee and Seung [NMF; 1999]) is that the sparseness of the computed activities Y can be set to a desired level. An increase in the sparseness shifts the code from global to more local features Hoyer [2004]. However, NMFSC deviates from a multiplicative update of the output Y and uses a subtractive one:

$$Y \leftarrow Y - \mu V^T(VY - X)$$

Thus, the nodes compete with each other using a top-down, subtractive inhibition of their input. To obtain the desired level of sparseness, a projection step is applied by keeping V fixed and looking for the closest Y which could both optimally cause to low reconstruction error and satisfy the sparseness constraint (for details see Hoyer [2004, pp. 1462-63]). NMFSC also allows controlling the sparseness of V.

Predictive coding/biased competition

In predictive coding/biased competition (Spratling [2010]), like in the two other mentioned generative models, the goal is finding components so that the output could resemble the input with minimal error. This method uses divisive input modulation (DIM), introduced in Spratling et al. [2009], which is in turn based on NMF. The modifications, in comparison to NMF, are mainly two. First, it is online, while NMF is a batch method. Second, in contrast to NMF which uses the component weight matrix both for computing the output and reconstructing the input, DIM considers two sets of weight matrices: feed-forward for producing the output and feedback for generating the reconstruction of the input. The two weight matrices differ just in the form of normalization, which makes the method stronger than NMF in the case of overlap and occlusion (Spratling et al. [2009]). In PC/BC, the inputs are inhibited by being divided by their reconstruction. This is done explicitly in the units called "error units". The error units basically

do the same job as the term $(XY^T) \oslash (WYY^T)$ in (Eqn. 2.1) in NMF. Their activity is described as following:

$$e = x \oslash (\epsilon_1 + \hat{W}^T y)$$

where x is the input vector, y is the output vector, $\hat{W}$ is the feedback weight matrix which is the same as W but normilized so that all rows of it have the same maximum value. ϵ_1 is a small value to avoid division by zero. The inhibited input from the error units is used for both producing the output and updating the weights. Thus, PC/BC uses in both cases a multiplicative updating, whereas NMFSC uses a subtractive one for the output.

In the calculation of the output, the inhibited input is used:

$$y \leftarrow (\epsilon_2 + y) \otimes We$$

where ϵ_2 is a random small number which prevents the output from being zero. W is the feed-forward weight matrix, and e is the activity vector of the error units. Based on the output activities y and the error units e the weights are adapted as following:

$$W \leftarrow W \otimes \{(1 + \beta y)[e^T - 1]\}$$

where β is the learning rate. If the input and its reconstruction are equal, the error will be equal to unity and, thus, the weights will not change.

Besides the weight development, the inhibition in PC/BC affects the output. Strong units suppress weaker ones by removing their representation from the input. This is done in several iterations of updating the error units based on the reconstruction of the output units. This iterative process leads to a low reconstruction error and provides the competitive mechanism of PC/BC.

Auto-encoders

Auto-encoders could be compared with one layer of neurons which are mirrored with respect to the output. That is, the output is fed to a similar network having the same number of dimensions on the input as the original network has on its output. The mir-

rored network has the same dimensionality on the output as the original network has on its input. The idea is that the original network and its mirror act as an encoder and decoder of information respectively (Figure 2.3). The optimization objective of auto-encoders is the minimization of the difference between the actual input and the decoded information after coding, which could be done using gradient decent Le et al. [2015].

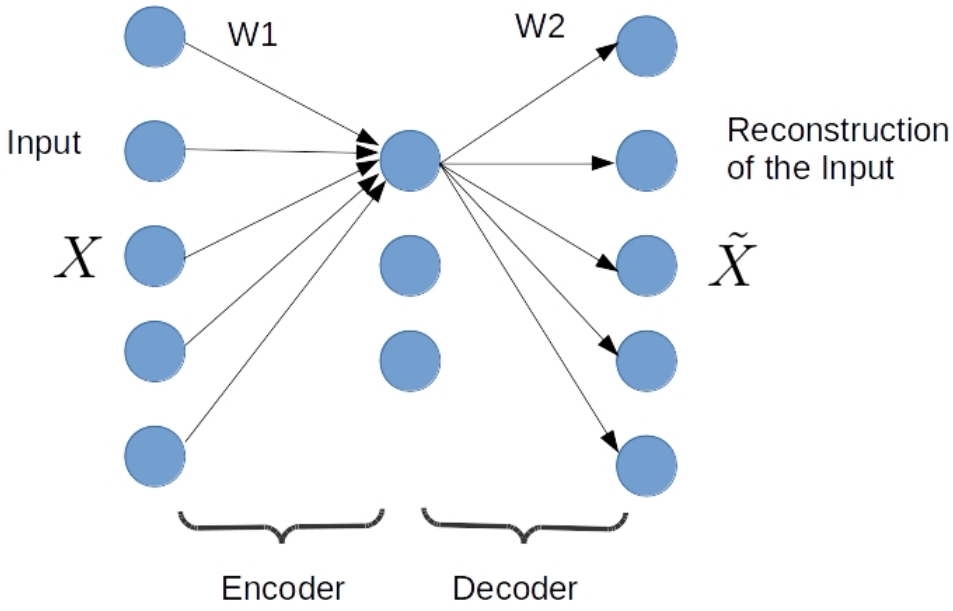


Figure 2.3: Autoencoder

Auto-encoders could be used for information compression or other purposes. In deep learning architectures a stack of Auto-encoders is used for information transformation before passing the output to a classifier. The procedure is to train layers of Auto-encoders one by one from bottom to top, then to add a classifier to the top. Then the entire network is fine-tuned in a supervised manner (using backpropagation). This architecture is called **Deep Auto-encoder**. An example of the application of layers of Autoencoders is the processing of financial news to find patterns which help us to predict the direction of stock market change (Fehrer and Feuerriegel [2015]).

Deep Restricted Boltzmann Machine

In order to understand Deep Restricted Boltzmann Machine, one should first under-

stand Restricted Boltzmann Machine (RBM). In contrast to most of generative models which are causal, Restricted Boltzmann Machine (RBM) (Salakhutdinov et al. [2007]) belongs to energy based generative models. In causal models, reconstruction of the input is the weighted sum of the corresponding output. In energy-based models we seek an equilibrium state which is satisfied when the energy of the joint configuration of the input and output is minimal. Here the model is generative in the sense that it produces a total distribution of the **input** and **output** which cause the minimal energy. The energy of the model is defined as follows:

$$E(V,h) = -\sum_{1=1}^{m}\sum_{j=1}^{F}\sum_{k=1}^{k} W_{ij}^{k} h_j v_i^k + \sum_{i=1}^{m} logZ_i - \sum_{i=1}^{m}\sum_{k=1}^{k} v_i^k b_i^k - \sum_{j=1}^{F} h_j b_j \quad (2.2)$$

Here $E(V,h)$ is the energy of a model with V and h being the sets of visible/input and hidden/output units; W_{ij}^k is the wight connecting the k^{th} unit of the i^{th} input to the j^{th} output. b_i^k is the bias of the k^{th} unit of the i^{th} input. $Z_i = \sum_{l=1}^{k} exp(b_i^l + \sum_j h_j W_{ij}^l)$ is a normalization term which guarantees that $\sum_{1=1}^{k} p(v_i^l = 1|h) = 1$.

In the learning phase, while feeding an input to the system, the system tries to find weights which minimize the energy of the system. Later on, in the evaluation/usage phase, under any new input, the system approaches the closest learned equilibrium state to the state the actual input causes. This way, the model will produce a similar output as if the optimal input had been present. This means such models are capable of compensating distortions in the input because of good generalization they do.

Restricted Boltzmann Machine is a simplified type of Boltzmann machine. Unlike the ordinary Boltzmann Machine (BM) which is a fully connected graph, in Restricted Boltzmann Machine (RBM) the connections are just between the visible (input) and hidden (output) units. Boltzmann machine is an improved version of Hopfield network Hopfield [1982]. The improvement is in the stochastic nature of the units in BM.

The stochastic nature of Boltzmann machine gives it the advantage of jumping out of local minima, although in Hopfield network, the search is only in the direction of decreasing energy which could cause the model to trap in local minima.

Learning in Boltzmann Machine is based on the Hebbian principle of learning; the

weight between two units is increased if the two units have correlation. Although, the weight reduction is not done by normalization of weights (Oja [1982]), but by Hebbian forgetting (Rojas [1996]). Weight increase and decrease are done in two different phases called **fixed** and **free**:

$$\Delta w_{ij} = \tau(\langle x_i x_j \rangle_{fixed} - \langle x_i x_j \rangle_{free}) \tag{2.3}$$

where τ is the learning rate and x_i and x_j are the states of different units. In the **fixed phase**, the visible units are clamped to the input and the weights between the visible and hidden units are increased based on the Hebbian rule. On the other hand, in the **free phase**, the weights are decreased as a function of the correlation between units in the absence of the input stimulus. That is, if in the free phase two units are often simultaneously active, the weight between them is decreased. This phase is called "unlearning" (Hopfield et al. [1983]). With unlearning we avoid several nearby local minima which could shape a high local minimum which is the aggregate of all minima in the neighborhood. A stack of RBMs is called a Deep Boltzmann Machine in which the hidden units of each lower layer are the visible units of the upper layer and the model is learned layer by layer.

2.2.2 Non-generative unsupervised learning

Although non-generative models are not as accurate and practical as generative ones, they are important in the study of our nervous system and are studied in computational neuroscience. The most famous model of this sort could be Hebbian leaning which tries to simulate the neural plasticity on a very local level and study how a single neuron or a neighborhood of neurons in the brain process the data.

Hebbian learning: Although natural systems approach some global goals, from the point of view of a neuron, everything is just about the local interactions. The ability of local and independent information processing is the reason neurons can act in parallel and although each of them is not as fast as a digital processor, they all together act faster than any computer. Understanding the Hebbian rule of learning which explains how the synaptic connections locally modify themselves as a result of the cellular interaction,

is essential in understanding the functions of the nervous system. Because of their generative and supervised nature, recent successful neural models have been far away from Hebbian learning, but, understanding the Hebbian as the most basic function of the nervous system could help to imagine its more complex functions and at the end lead to developing more efficient methods for learning and processing.

Hebbian learning is based on the idea introduced by Hebb [1949] which suggests that the neurons which fire together wire together, which means, the synaptic weight between two neurons implies the correlation between the neurons:

"When an axon in cell A is near enough to excite cell B and repeatedly and persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency in firing B, is increased" (Hebb [1949]).

One could mathematically express this idea as the following equation:

$$\tau \frac{\partial w}{\partial t} = r^{pre} \cdot r^{post}$$

that is, the strength of the synaptic weight w from the pre-synaptic neuron with the activity r^{pre} to the post-synaptic neuron with the activity r^{post} increases proportional to the strength of the product of these activities. The speed of increase is controlled by the learning rate τ.

A neuron could be considered as a decision maker who decides to which extent an input pattern is toward the direction of the weight vector it has learned. After the learning is complete, if a neuron observes a pattern which matches its learned receptive field, it will show a higher activity than other neurons which have learned other patterns. The reason is that the inner product of two vectors is maximum if the Euclidean distance between them is minimum (Figure 2.4). One should pay attention that this is correct just if the vectors have equal length. Considering the input patch and the weight matrix as two vectors, the output of the neuron is actually a function of the sum of the elements of the inner product of these two vectors.

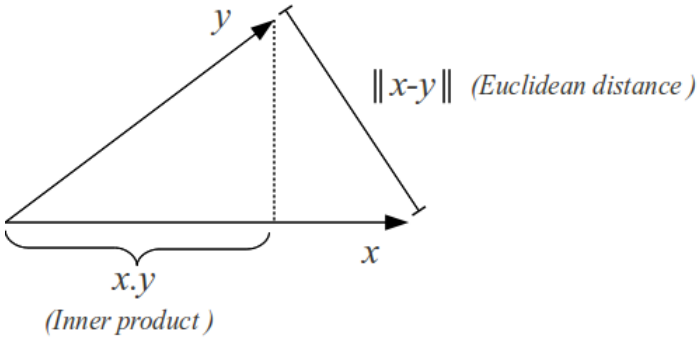


Figure 2.4: Euclidean distance between two vectors

The Hebbian learning rule has the problem that the weights increase unbounded. Oja improved this learning rule (Oja [1982]) by adding a weight reduction coefficient α to the learning rule:

$$\tau \frac{\partial w_{ij}}{\partial t} = (r^{pre} \cdot r^{post}) - \alpha {r^{post}}^2 w_{ij}$$

It can be proven that this modification is equal to normalization of the weight vectors by devision which prevents the weight vector from unbounded increase. In Oja [1982] Oja proved that this method leads to finding the first principle component of the input. This is since his learning rule seeks the directions in which the second order statistics, that is the variance of the input is maximized.

The Hebbian rule, explains Long Term Potentiating but is not capable for explaining Long Term Depression. In an alternative implementation, by forcing the network to decrease the activity if the input or output is lower that a threshold, we can simulate Long Term Depression too:

$$\tau \frac{\partial w}{\partial t} = (r^{pre} - \theta_{pre})(r^{post} - \theta^{post})$$

where θ^{pre} is the mean value of post-synaptic activations and θ^{post} the mean value of the pre-synaptic activities. This method which is similar to calculating the covariance between r^{pre} and r^{post} is called Covariance Learning (Bell and Sejnowski [1997]).

One point in this method is that, in the case of low input and output $((r^{pre} < \theta_{pre}) \wedge (r^{post} < \theta^{post}))$ the result will be an undesired increase in w. So in practice one could

use:

$$\tau \frac{\partial w}{\partial t} = (r^{pre})(r^{post} - \theta^{post})$$

,

$$\tau \frac{\partial w}{dt} = (r^{pre} - \theta_{pre})(r^{post})$$

or

$$\tau \frac{\partial w}{\partial t} = (r^{pre} - \theta_{pre})^{+}(r^{post} - \theta^{post})$$

Another Hebbian based learning rule was proposed by Bienenstock et al. [1982] in which the synaptic efficacy of active inputs increases when the post-synaptic target is higher than a modification threshold θ. On the other hand, when the level of post-synaptic activity falls below θ, the strength of active synapses decreases. In Bienenstock et al. [1982] Bienenstock, Cooper and Munro (after whom the method is called BCM) proposed that the value of the modification threshold should not be fixed, but should vary as a nonlinear function of the average output of the post-synaptic neuron.

$$\tau \frac{\partial w}{\partial t} = \Phi(r^{post}, \theta) \cdot r^{pre}$$

where $\Phi(r^{post}, \theta)$ is a scalar function of the post-synaptic activity r^{post}, that changes sign at a threshold, θ.

The vector w is driven in the direction of the input r^{pre} if the output is large (above θ), or opposite to the direction of the input if the output is small (below θ). There are several possible mathematical forms of the Φ function (Intrator and Cooper [1992]), although the original form presented in BCM is:

$$\Phi(r^{pre}, \theta) = r^{post} - (r^{post})^2$$

Competitive learning and sparse coding: The learning rules and other mechanisms used in this thesis are inspired from several works which are explained in this section. They are all unsupervised models following the Hebb's learning rule for learning.

We start from a very important paper, Foldiak [1990], which is the base of the inhibitory mechanism used in this work. Foldiak [1990] introduced a competitive learning method (von der Malsburg [1973],Grossberg [1987]) for sparse coding, based on Hebbian and anti-Hebbian learning. Sparse coding as Foldiak [1990] states, is a compromise between local and distributed feature learning. By local, we mean a feature learner with the capacity of learning just a few but highly decorrelated features. On the other hand, distributed representations may learn many overlapping features which cover a big portion of the space, but produce a dense and redundant code. As Foldiak [1990] describes, while components of objects could be highly correlated, the objects themselves are almost independent of each other. Each object could be understood as groups of dependent (correlated) features which are independent from other features (Foldiak [1990]). In his simplified one-layer network, the units are not learning objects but components which shape the objects. Each unit or neuron learns some sort of dependency between the pixels of images through Hebbian learning while the ani-Hebbian learning tries to decorrelate the units guaranteeing that the components are independent of each other. Foldiak [1990] states that the goal of nervous system could be finding these groups of features as explicit as possible and finding a reduced representation with minimal redundancy.

The introduced model by Földiak benefits from modifiable inhibitory weights which let a soft competition in which more than one unit could be active; thus, is not a pure winner-take-all (WTA) method. Another characteristic of this model is the usage of an adjustable sensitivity threshold to keep neurons active. In this model, each input is fed to the network for several steps because the final output is influenced by the inhibition and the activities cannot be calculated in one step. Figure 2.5 and equation 2.4 show architecture and formulation of this method.

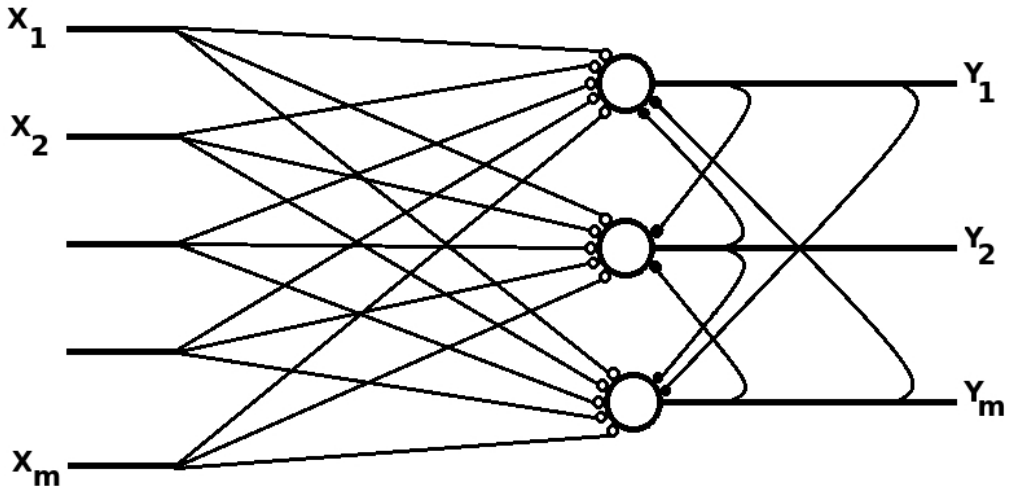


Figure 2.5: Hebbian and anti-Hebbian weights in Foldiak [1990]

$$\frac{dy_i{}^*}{dt} = f(\sum_{j=1}^{m} q_{ij}x_i + \sum_{j=1}^{n} w_{ij}y_j{}^*) - y_i{}^* \tag{2.4}$$

where q_{ij} are the feed-forward weights, x_i the input value, w_{ij} the feed-back inhibitory weights $y_j{}^*$ a function of the output y_j:

$$y_i{}^* = \begin{cases} 1 & if \quad y_i \geq 0.5 \\ 0 & otherwise \end{cases}$$

$$f(u) = \frac{1}{1 + exp(-\lambda u)}$$

In line with the Oja's PCA algorithm (Oja [1982]) one could refer to a more complex work named **Generalized Hebbian Algorithm** (GHA) by Sanger [1989]. This work is based on a gradient decent algorithm with back-propagation (Rumelhart et al. [1986]) and PCA. In this work, a Hebbian learning which finds the eigenvectors of the input is employed. The Generalized Hebbian Algorithm is a combination of the Oja's learning rule which is capable of finding just the first PC and the Gram-Schmid orthogonalization

process which causes the decoration of the output signals. The weights of the network demonstrate the first few eigenvectors of the autocorrelation matrix of the input in the descending order. This is achieved by subtracting the activity of the first neuron from the input of the second neuron, the activities of the first and second neurons from the input of the third neuron and so on. By subtracting the activities of the previous neurons which are the previous orthogonal principal components, we let the current neuron to learn the next direction of the data which increases the variance. That is, to learn the next and the less valuable principle component of the data (Figure 2.6 and equation 2.5):

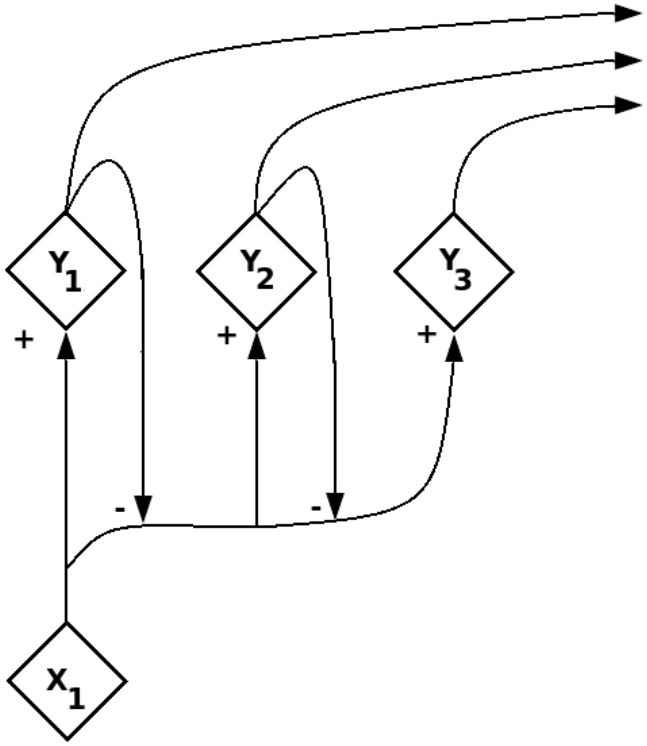


Figure 2.6: Generalized Hebbian Algorithm. The outputs of the previous units are subtracted from the input to guarantee decorrelation.

$$x_j{}'(t) = x_j(t) - \sum_{k<j} c_{kj}(t) y_k(t) \tag{2.5}$$

where c_{kj} are the feedback inhibitory weights from the previous units.

Later on, Olshausen and Field [1996] claim that an unsupervised method with two objectives: a) maximizing sparseness of the neural activities and b)minimizing the reconstruction error would produce a complete set of spatially localized, oriented and bandpass basis functions (filters) which mimic the receptive fields of the simple cells in V1 layer of the visual cortex. The basic idea is that an observed image could be represented as a linear superposition of the basis functions learned by the cell population.

$$I(x,y) = \sum_i a_i \Phi_i(x,y) \tag{2.6}$$

where the a_i coefficients are the output value of a cell when observing the current image, Φ is the receptive filed of the cell which is shown as a basis function of pixels in the x-y surface of the input image I.
The objective is minimizing the following cost function:

$$E = (\sum_{xy}[I(x,y) - \sum_i a_i \phi_i(x,y)]^2) - (\lambda \sum_i S(\frac{a_i}{\sigma})) \tag{2.7}$$

in which the first part is the reconstruction error and minimizing the second part will cause maximizing sparseness. Here S is a non-linear function like $-e^{x^2}$, $log(1+x^2)$ or $|x|$ which favor activity populations where the number of non-zero activities is higher, or in other words, the sparseness is higher. σ and λ are normalization factors.

Looking for the interesting components or basis functions which could explain the image the best, this algorithm seeks a set of Φ which, while maximizing the sparseness of the neural activities, minimizes the reconstruction error. One could think of the Principal Component Analysis as the base of this method, but here we achieve over-complete basis functions instead of orthogonal ones provided by PCA. This helps to describe an image when slightly different components are needed, or the components somehow overlap.

Rehn and Sommer [2007] introduced a "hard sparseness" in which instead of keeping the mean activity low (soft sparseness), the number of active neurons is limited. The idea is that the receptive fields should correspond to statistically independent components of natural images. Supposing that in a single patch of natural images, the number of this statistically independent components is limited, a small number of neurons should

respond to it.

A later work, Falconbridge et al. [2006], was an attempt to make a biologically plausible model inspired by the Hebbian/anti-Hebbian sparse coding algorithm (Foldiak [1990]). It was shown that the modified version of the algorithm is capable of extracting the sparse and independent components of natural images. The model is modified so that unlike the Foldiak [1990] model which produced binary output, could produce continuous activities. The other modification to the model was the usage of negative and positive input values inspired from the on and off LGN activities. The output are calculated as follows:

$$y_i{}^q = f(\sigma_i{}^q + \sum_j u_{ij} y_j{}^q + S_i) \tag{2.8}$$

where $\sigma_i{}^q = \sum_j w_{ij} x_j{}^q$ is the weighted sum of the input and S_i is the sensitivity threshold mdified this way: $\Delta S_i = \eta(\alpha - y_i)$ in which η is the learning rate and α the desired average activity of all output units. u_{ij} are the lateral weights connecting the output unit $y_i{}^q$ to each other.

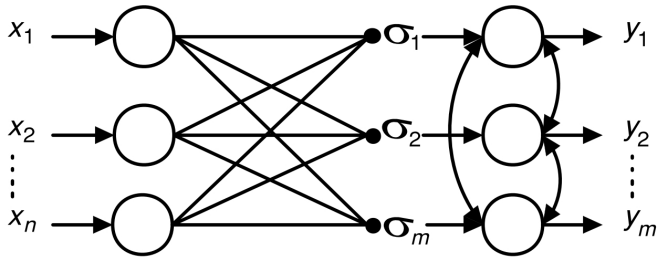


Figure 2.7: The Hebbia/anti-Hebbian network introduced in Falconbridge et al. [2006]

A more advanced model was introduced in Hamker and Wiltschut [2007]. This model is a non-generative model although with feed-back weights which are learned simultaneously with the feed-forward weights (Figure 2.8). The job of feed-back weights, unlike in generative models, is match-enhancement based on the activities of the neurons. If a weak input causes high activities, it will be reinforced through feedback weights (equation 2.9). This will result in weight increase based on the Hebbian rule of learning which is a function of the multiplication of the input and the output.

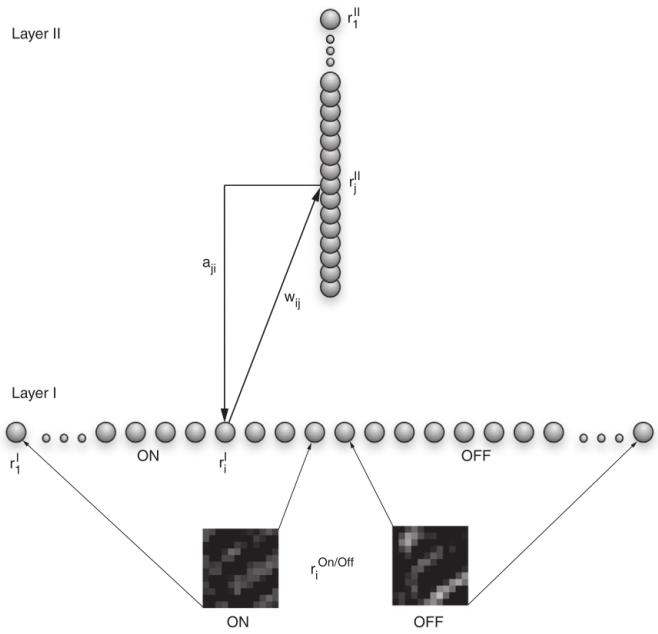


Figure 2.8: A two layer neural network model used in Hamker and Wiltschut [2007] with feed-forward and feed back connections simulating simple cells of the primary visual cortex.

$$\tau \frac{dr_i{}^I}{dt} = r_i{}^{On/Off}(1 + (\lambda - \max_x \quad r_k{}^k)\sum_j a_{ij} r_j{}^{II}) - r_i{}^I \tag{2.9}$$

The diversity of filters is guaranteed through pre-synaptic inhibition which does not let several neurons to be simultaneously active. This is achieved by suppressing the input with multiplying it to a function of post-synaptic activities and their corresponding feed-forward weights (equation 2.10). This mechanism suppresses the input in the case the current pre-synaptic activity strongly stimulates another neuron.

$$\tau \frac{dr_j{}^{II}}{dt} = \sum_i [w_{ij} r^I{}_i (\max_{(k,k\neq j)} (\frac{w_{ik}}{max_m w_{mk}} \frac{r_k{}^{II}}{max_n r_n{}^{II}}))^+] - r_j{}^{II} \tag{2.10}$$

Wiltschut and Hamker [2009]: This work is similar to Hamker and Wiltschut [2007] with the difference that the second layer neurons receive non-linear self-excitation and non-linear lateral inhibition (figure 2.9 and equation 2.11).

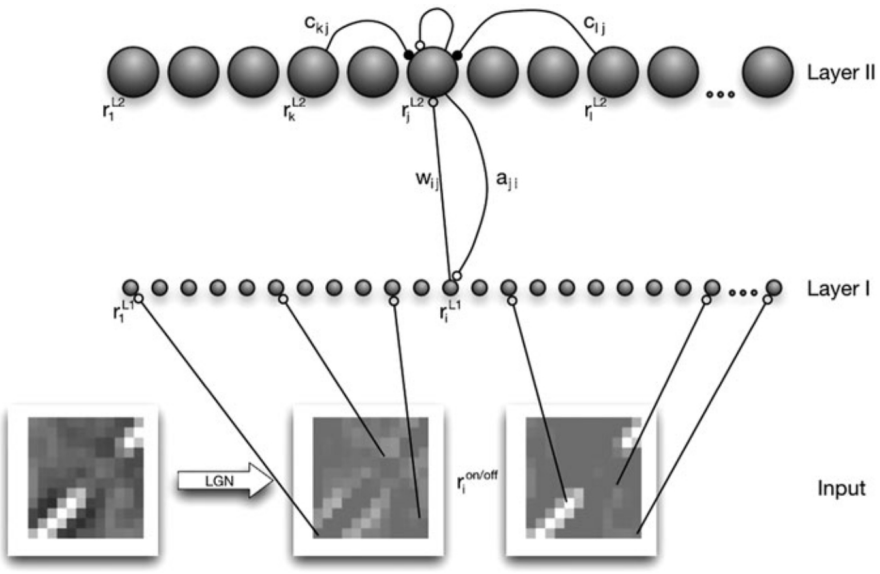


Figure 2.9: The two layer model used in Wiltschut and Hamker [2009]. The attentional feed-back signal modulates the input on the first layer. The second layer simulates the V1 cells. The cells of the second layer compete using lateral inhibition.

$$\tau_{L2}\frac{\partial {r_j}^{L2}}{\partial t} = \sum_i w_{ij}{r_i}^{L1} + \eta.f(s^{L2}.{r_j}^{L2}) - \sum_{y,y\neq j} f(c_{jy}{r_y}^{l2}) - {r_j}^{L2} \tag{2.11}$$

where

$$f(x) = d_{nl}.log(\frac{1+x}{1-x})$$

and

$$x^{L2} = (A_{L2} - max_y {r_y}^{L2})$$

where $\eta = 2$, $A_{L2} = 0.9$ and $\tau_{L2} = 10$

Teichmann et al. [2012]: This work is the main basis of the model used in the first part of the practical work of this thesis (Chapter 3). The focus of Teichmann et al. [2012] is on the achievement of simple and complex cells with differing the calcium level which affects the speed with which a neuron's membrane potential follows the weighted sum of the inputs. The idea is that with a lower speed, some shift invariance which is specific

to complex cells will be achieved if an input is slightly shifted and repeatedly fed to the network. In Chapter 3 of this thesis though, the focus is not on the shift invariance and a fixed calcium level for simple cells is used. The main idea taken from Teichmann et al. [2012] is the learning rules and specifically the adaptive weight reduction factor for feed-forward weights which are explained later.

2.3 Competition

In order to have several units detecting different features, a mechanism should prevent them from learning the same pattern. One solution to this, is competitive learning. In the simplest form, competitive learning is a winner-take-all mechanism in which just the units which best match the input are allowed to learn (Rumelhart [1985]). Another example of inducing competition is Dropout (Srivastava et al. [2014]) in which under presentation of each stimulus, a half of feature detectors are randomly turned off to prevent co-adaptation of feature detectors. This is helpful in the case the number of feature detectors is relatively large and prevents over-fitting. In a similar method called Drop Connect, each unit receives signals from a random subset of the units in the previous layer. This is done by setting a randomly chosen subset of weights to zero (Wan et al. [2013]).

An influential work based on Rumelhart [1985] which attempts to understand the function of inhibitory mechanism in the neural system is Foldiak [1990] which explains inhibitory connections as anti-Hebbian weights that in contrast to Hebbian connections are redundancy detectors. The task of these Anti-Hebbian connections is reducing the statistical dependency among neurons. In Foldiak [1990] model, the units are connected with feedback inhibitory weights which grow in the case of correlation between the units and de-correlate the units.

There exist some studies showing that applying inhibition to some well-known neural models can improve their performance. Some examples of those works are:

- Fukushima [1980], Fukushima [1988] and Perez et al. [2003] added inhibitory mechanisms to models based on Neocognitron to improve its performance.

- Tivive and Bouzerdoum [2005] added inhibition to LeCun's Convolutional Neural Network to enhance its classification performance.

- Mao and Massaquoi [2007] showed that adding lateral inhibition to a recurrent neural network proposed by Ution [1992] increases its robustness and performance.

- Fernandes et al. [2013] employs lateral inhibition in PyraNet proposed by Phung and Bouzerdoum [2007] to bring stability to the network and improve its performance in texture analysis.

- Another interesting example of improvement of existing models by adding inhibition is Mutch and Lowe [2006] in which a deep learning mechanism is equipped with an inhibitory mechanism. In this model, lateral inhibition is used to increase sparsity which prevents from learning a combination of features by favoring the dominant ones. This model is an extension to H-MAX model (Riesenhuber and Poggio [1999]) and shares a lot with Neocognitron and Convolutional neural networks. Like in Neocognitron and CNN, Mutch and Lowe [2006] also has layers of feature extractors which compute the dot product (convolution) of the input to the features and compute higher order features. Each of these layers is followed by a pooling layer which brings invariance to the system. The first layer in this model is a pre-designed Gabor filter set.

2.3.1 Pooling

Pooling and inhibition share the same idea in the sense that both try to omit a part of the information to achieve abstraction and to avoid redundancy. The difference, however, is that in inhibition, redundancy is avoided by forcing units to learn different features, but in pooling, redundant information is avoided by grouping neighboring units and considering them as a single unit representing one particular feature.

In some deep learning architectures, after each layer of feature detectors, a layer of pooling units is used to bring invariance and robustness to the system. Each pooling unit combines the outputs of a group of neighboring units from the lower layer. This combination, named pooling, could be done in different ways. Of the most popular pooling techniques, one could count sum-pooling, max-pooling, average-pooling, feature pooling and Generalized Max-Pooling Murray et al. [2014]. Sum-pooling is not

strongly discriminative because it is highly influenced by frequent features that may not be always informative, but not sensitive to less frequent features which may be more important. Max-pooling, on the other hand, equalizes the effect of rare and frequent features but it doesn't consider the probable correlation between dimensions and considers them independently. To overcome this shortage, Generalized Max-Pooling (GMP) tries to reduce the reconstruction error of each patch after pooling Murray et al. [2014]. Famous examples of pooling are Fukushima and Miyake [1982] with feature pooling, LeCun [1989], LeCun et al. [1998] and Pinto et al. [2008] with average-pooling, Ranzato et al. [2008], Jarrett et al. [2009] and Serre and Poggio with max-pooling. Alternatively, pooling could be also learned (Teichmann et al. [2012]) in a more biologically plausible manner by modifying the speed of change in the output activity. In Teichmann et al. [2012] this was inspired by the effect of different calcium levels on the characteristics of learning.

Pooling has its biological roots in the idea of **complex cells** introduced by Hubel and Wiesel [1982] and its analytical root in the concept of **locally orderless images** introduced by Koenderink [1999]. For a more recent work about the relation between complex cells in the visual cortex and pooling, one could refer to Boureau et al. [2010].

The concept of locally orderless images could be explained as follows: if we replace the pixel information of a region of interest in an image with the histogram of intensities, we are removing the order from the calculation and thus making a locally orderless image. This could bring some level of invariance to image processing in tasks like texture detection. As an example, histograms of black and white stripes with different thicknesses are distinguishable from each other, but histogram of horizontal and vertical images are not; that is a level of rotation invariance could be achieved just by replacing the pixel data with the histogram (Figure 2.10).

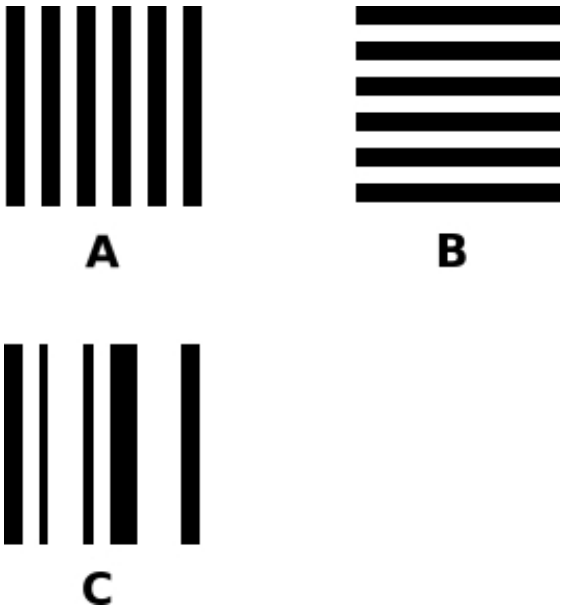


Figure 2.10: Replacing the pixel information of some areas of images with their histogram may bring invariance. The histograms of image A and B would be the same, but with high probability different from the histogram of the image C. That is, the histogram here is invariance to rotation.

One important factor in pooling is to try aggregating features without losing information (Boureau et al. [2011]). Although pooling is done in a neighborhood of features, there is no guarantee that the neighboring features are similar and pooling could cause severe loss of information. Here the idea of feature maps and weigh sharing comes up. By pooling over similar features (in a feature map), we could minimize the loss of information under pooling (Boureau et al. [2011]).

In an example of the convolutional neural network (Abdel-hamid et al. [2013]) the authors use a soft-max pooling in which the pooling is done using learnable weights. The pooling weights are updated based on derivative of the error with respect to the soft-max of the activities of the convolutional layer. The authors claim that this is equal to having the dynamic size for pooling windows. The performance is comparable with that of fixed-size max-pooling and showed improvement.

The popular 2*2 max-pooling (MP2) used in CNNs applies max-pooling to 2*2 windows and as a result, the reduction in the size of the layers is very quick. Although this will bring invariance, the loss of useful information could be too much. Some works attempt to overcome this problem by using overlapping filters (Krizhevsky et al. [2012])

or stochastic pooling (Zeiler and Fergus [2013]). In a recent pooling method, Fractional Max-Pooling (FMP), this problem is solved by seeking the best window size for max pooling. It is shown that if the window size is a $\sqrt{2}$ instead of 2 and the filters do not overlap the performance will increase (Graham [2015])

Chapter 3

Competition improves robustness against loss of information

In this chapter, the role of competition among the neurons of a single layer neural network in overcoming confusion in the case of occlusion on the input is investigated. Since typical forms of perturbations locally affecting V1-cells can be different lightning conditions like reflections or flares; unclear media like soiled glasses, windows, heated air; or covered objects like the view through a fence, occlusion used in this work was in the form of random removal of parts of visual data.

Because of the simplicity of measurement and experiments, the visual cortex has been traditionally a target for studying how the brain processes the information received from the environment. Several different learning approaches have been developed to model early vision, particularly at the level of V1 (Olshausen and Field [1996], Bell and Sejnowski [1997], Hoyer and Hyvärinen [2000], Falconbridge et al. [2006], Rehn and Sommer [2007], Wiltschut and Hamker [2009], Spratling [2010], Zylberberg et al. [2011]). In many of the works, the proposed characteristics of the visual system have been considered as optimization objectives and thus as criteria for measuring the efficiency of coding. Several kinds of optimization objectives, like sparseness of activity (Olshausen and Field [1996], Hoyer [2004]) or independence (Bell and Sejnowski [1997], van Hateren and van der Schaaf [1998]) have been used for this purpose. The major criteria for evaluation of those models are the ability to develop localized, oriented, bandpass receptive fields and the similarity of the distribution of receptive fields

to the observed ones in the macaque (Ringach [2002]). Because of high complexity and non-observability of the neural system, although visual or statistical characteristics of a model are proper descriptors of the correctness of a model, they are not sufficient or not always informative. One way to overcome this limitation is studying the ability of the models in solving practical problems in which the brain is involved. In the current study, classification of handwritten digits from famous MNIST dataset is considered as the criterion for the efficiency of the models.

The first step in the hierarchy of natural image processors is detecting edges which is the responsibility of cells in V1 (Hubel and Wiesel [1982]). It is shown that this natural edge detectors resemble Gabor filters (Olshausen and Field [1996]). Gabor filters are capable of detecting edges with particular frequencies, size, and orientations. A detailed practical explanation about the Gabor filters could be found in appendix section Gabor Filter. The neural network used in this chapter for extracting features from digit images was initially built to simulate V1 layer of the Visual Cortex and to learn the receptive fields based on natural images (Teichmann et al. [2012]). When fed by natural images, the network is capable of learning Gabor like filters (Hamker and Wiltschut [2007]) which mimic the receptive fields of the V1 layer (Figure 3.1). This similarity is an evidence of the fact that the brain at least to some limited extent tries to learn the statistics of the environment for better interpretation of the surrounding world through an unsupervised learning mechanism. After the first layer, V1, transfers the visual data to a new representation by detecting the edges, the higher layers can easier interpret the data. This principle was used in the current study to simplify the recognition of digits after they are transferred to a new representation in the form of neural activities.

For a deeper understanding of the role of interactions between network units, three methods implementing different learning strategies were used for feature detection: Fast independent component analysis (FastICA) (Hyvärinen and Oja [1997], Hoyer and Hyvärinen [2000]), non-negative matrix factorization with sparseness constraint (NMFSC) (Hoyer [2004]), and Predictive Coding/Biased Competition (PC/BC) (Spratling [2012]). FastICA was chosen as a method which tries to find new representations of the data with the minimal dependency between components without employing any kind of competition. Non-negative matrix factorization introduces competition among components by inhibiting the input (Spratling et al. [2009]). NMFSC also puts an additional

constraint on the output of the system and keeps it sparse on a desired level. Predictive Coding / Biased Competition (Spratling [2012]) tries to reduce the difference between the input and its reconstruction divisively and employs feedback inhibitory connections. All the mentioned methods and the Hebbian neural network introduced in this work, are capable of learning V1 simple-cell like receptive fields from natural images.

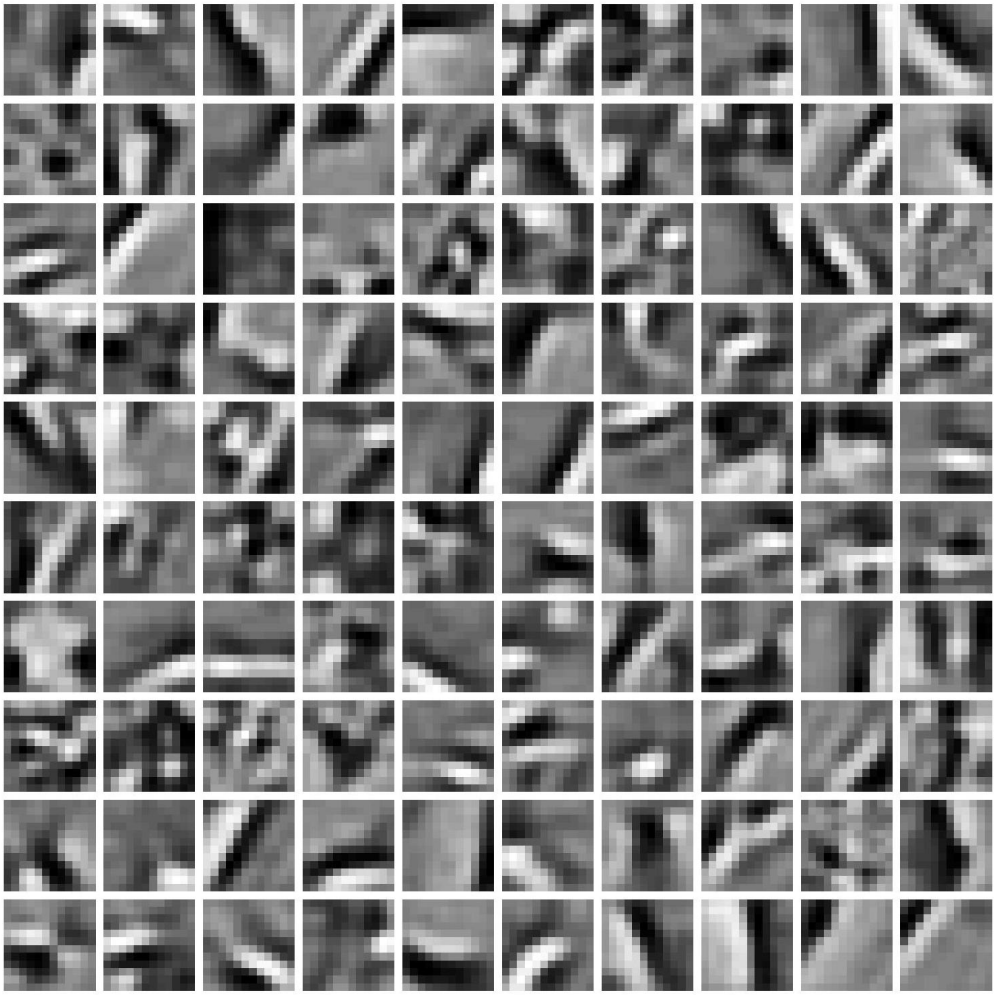


Figure 3.1: Receptive fields learned from natural images resemble Gabor filters and act as edge detectors

3.1 Materials and Methods

3.1.1 Occlusion

In order to test the effect of loss of information on classification, some percentage of the non-zero pixels of the input images were randomly set to zero (Figure 3.2). The occlusion was applied to the test images from 0 to 60 percent.

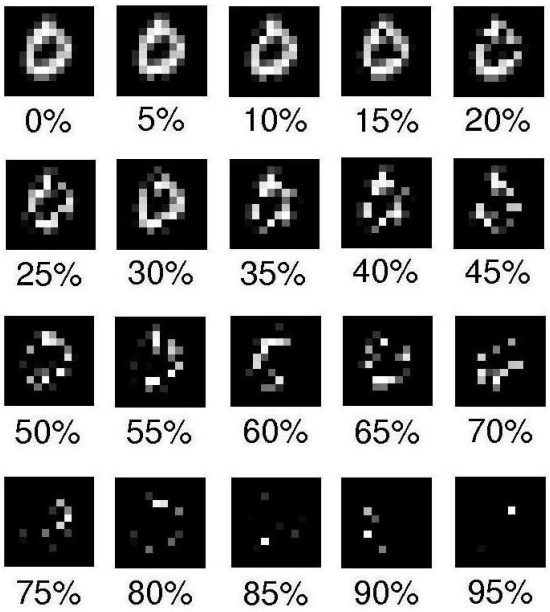


Figure 2. An example of 0% to 95% occlusion of the input digit patches.

3.1.2 Input preprocessing

As the network was originally configured to receive 12×12 patches of natural images Wiltschut and Hamker [2009], Teichmann et al. [2012], the MNIST digits were resized from 28×28 to 12×12 pixel patches using the MATLAB function imresize() by the factor of 0.40. In order to simulate the function of the early visual system up to the Lateral Geniculate Nucleus (LGN), transferring the signals from the eyes to V1, the images were whitened using the same method as Olshausen and Field [1996] (see also Wiltschut and Hamker [2009]). The whitening leads to positive and negative values. The positive part reminds us of the so-called on-center receptive fields of the LGN cells and the negative part, the off-center receptive fields Wiltschut and Hamker [2009].

Because of biological plausibility, the input to the network cannot be negative. So, each whitened image which is actually a matrix was separated to two matrices with the same size; one containing the positive and one containing the negative values of the original weight matrix. The negative matrix was then negated to have positive values proper to be fed to the neural network.

3.1.3 The Hebbian/anti-Hebbian neural network

A biologically plausible neural network model employing Hebbian and anti-Hebbian plasticity mechanisms was used for feature extraction from pixel data. An earlier version of the model, Wiltschut and Hamker [2009], has shown its capability to learn Gabor-like receptive fields comparable to physiological data. The version used here was previously published in Teichmann et al. [2012] for the purpose of learning V1 complex-cell receptive fields. Some minor modifications to the model were introduced which are described in this section. The learning rule was applied to the feed-forward connections from the input layer to the ouput layer and to the lateral inhibitory connections among the neurons on the output layer. The lateral inhibitory connections are the source of competition in the network and facilitate the de-correlation of the neurons. After training, the lateral connections induce competition between the neurons based on the persistent correlations between the neurons in the learning phase.

In the learning phase, for 200,000 times, a digit from the set of 60,000 MNIST train set was randomly chosen. Each of these randomly selected digits was 100 times fed to the network repeatedly. After each five images belonging to a digit class, the class was randomly changed. After training, the weights were saved and later used for calculating the output on the test set.

3.2 The Neural Network model

In this section, the mathematical model of the neural network is explained. First, the modeling of the membrane potential and the activation function of neurons are introduced. Then the feed-forward weights which learn the receptive fields of the feature detector with a Hebbian like rule are explained. At the end, the competition among neu-

rons with the help of lateral weights between neurons of one layer is discussed. These weights are learned based on an anti-Hebbian method.

3.2.1 Neural activity

The membrane potential of the neurons was calculated by the weighted sum of the inputs and decreased by the amount of inhibition from the other neurons:

$$\tau_m \frac{\partial m_j}{\partial t} = \sum_i w_{ij} r_i^{pre} - \sum_{k,k\neq j} f\left(c_{kj} {r_k}^{post}\right) - {r_j}^{post} \tag{3.1}$$

with the nonlinearity function

$$f(x) = d_{nl} \cdot \log\left(\frac{1+x}{1-x}\right)$$

where m_j is the membrane potential of neuron j, w_{ij} is the weight of the pre-synaptic neuron i to the post-synaptic neuron j and c_{kj} is the inhibitory weight of a neuron k to the neuron j from the same layer. The constant $d_{nl} = 0.8$ is a modulation factor for the nonlinearity function and $\tau_m = 10ms$ is the time constant for temporal dynamics.

The activity of a neuron is a function of its membrane potential (Teichmann et al. [2012], Wiltschut and Hamker [2009]):

$$r_j = \begin{cases} 0.5 + \frac{1}{1+e^{-3.5(m_j-1)}} & m_j > 1 \\ (m_j)^+ & m_j \leq 1 \end{cases} \tag{3.2}$$

which prevents negative activities and keeps the activity between 0.0 and 1.5.

Learning Feed-Forward weights: The receptive fields are learned unsupervised with a Hebbian like algorithm introduced in Teichmann et al. [2012]. For learning the feed-

forward weights a "calcium level based" Hebbian rule was used:

$$\tau_{Learn,k}\frac{\partial w_{jk}}{\partial t} = \begin{cases} \left(\tilde{Ca}_j^{pre}\right)\left(\tilde{Ca}_k^{post}\right) - \left[C_{jk}^{alpha}\right]^{\Phi} & Ca_k^{post} > \bar{Ca}^{post} \\ \left(\tilde{Ca}_j^{pre}\right)^{+}\left(\tilde{Ca}_k^{post}\right) & \frac{1}{10}\bar{Ca}^{post} < Ca_k^{post} < \bar{Ca}^{post} \\ 0 & else \end{cases} \tag{3.3}$$

Where

$$X^{+} = \begin{cases} X & X > 1 \\ 0 & else \end{cases} \tag{3.4}$$

Here Ca_j indicates the activity in the neuron j. $\tilde{Ca}_j^{pre}$ is the distance of the current activity of the pre-synaptic neuron from the mean activity over all layer population and $\tilde{Ca}_j^{post}$ respectively is the distance of the current post-synaptic activity from the mean post-synaptic activity. This separates long term potentiation (LTP) from long time depression (LTD) as was introduced in BCM (Bienenstock et al. [1982]). The learning rule is not based on the immediate value of the input. Instead, the pre-synaptic activity r_i^{pre} is converted to a trace with a $\tau_{Ca,layer} = 10$. This learning rule follows the idea that learning rate in neurons depends on the level of calcium in the inter-cellular environment:

$$\tau_{Ca,layer}\frac{\partial Ca_i}{\partial t} = r_i - Ca_i \tag{3.5}$$

Similar to Oja's learning rule (Oja [1982]), the increase of the weights should be controlled. In our case, as the weights can be negative too in order to mimic the feed-forward inhibition, the decreasing factor is acting slightly different from the original

Oja's rule and also prevents infinite decrease of the weights:

$$\left[C_{jk}^{alpha}\right]^{\Phi} = \begin{cases} C_{jk}^{alpha} & \operatorname{sgn}\left(C_{jk}^{alpha}\right) \equiv \operatorname{sgn}\left(\tilde{Ca}_{j}^{pre} \cdot \tilde{Ca}_{k}^{post}\right) \\ 0 & else \end{cases} \tag{3.6}$$

where

$$C_{jk}^{alpha} = \alpha_k \left(\tilde{Ca}_{k}^{post}\right)^2 w_{jk} \tag{3.7}$$

$$\tau_\alpha \frac{\partial \alpha_k}{\partial t} = -\alpha^- + H_k \tag{3.8}$$

$$\alpha_k = (\alpha_k)^+$$

Where H_k is following half rectified membrane potential with the time constant $\tau_H = 100ms$:

$$\tau_H \frac{\partial H_k}{\partial t} = \left((m_k)^+\right)^2 - \frac{1}{Nneurons} - H_k \tag{3.9}$$

with

$$H_k = (H_k)^+$$

H_k is constantly decreased by a small value $\frac{1}{Nneurons}$ (with $Nneurons$,the number of the neurons in the layer) while increasing by the square of the membrane potential.

Dynamic learning rate for feed-forward weights

The learning rate is dynamically calculated based on the neural activity. When a neuron is consistently active, its learning rate increases, making it more sensitive and

more capable of learning. This way the neurons which have learned a pattern much different from the current input are less sensitive and the ones which detect the current input pattern as a familiar pattern are faster in learning:

$$\tau_{Learn,k} = a + b \cdot e^{\left(-c \cdot C a_k^{post}\right)} \tag{3.10}$$

Here $a = 5000$ is the lowest time constant (leading to the highest learning rate) and $b = 30000$ added to a defines the highest time constant (leading to the lowest learning rate). $c = 10$ indicates the decay of the exponential.

Lateral inhibitory weights For the lateral weights c_{kj} between the neurons an anti-Hebbian rule was used:

$$\tau_{c,k} \frac{\partial c_{kj}}{\partial t} = r_k^+ \cdot r_j^+ - \alpha_c \cdot r_j^+ \cdot c_{kj} \tag{3.11}$$

and

$$c_{kj} = (c_{kj})^+$$

where

$$\tau_{c,k} = a - a \cdot b \cdot e^{\left(-c \cdot r_k^+\right)} \tag{3.12}$$

Inhibitory weights de-correlate the outputs of neurons and lead to learning different receptive fields (Teichmann et al. [2012]). Each neuron has inhibitory weights to all other neurons. The amount of inhibition from other neurons is calculated by multiplying and summing up the activities of other neurons to the corresponding inhibitory weights connecting the actual neuron to the other neurons. This way if at the same time with the actual neuron some other neurons have high activities, meaning being sensitive to the actual input pattern, they reduce the membrane potential of the actual neuron with the help of inhibitory lateral weights and prevent it from learning. In other words, complete with the actual neuron.

3.2.2 fast Independent Component Analysis on MNIST

After the weight matrix W (section 2.2.1 for details about fastICA) was determined on the (non-occluded) MNIST train set, W was used to calculate the output on the occluded test set by calculating $y_o = Wx_o$, where x_o stands for the occluded input and y_o for the corresponding output activities. The FastICA method has no competitive mechanism affecting the output, it is just applying a linear transformation matrix on the input.

3.2.3 Non-negative matrix factorization with sparseness constraint (NMFSC) on MNIST

To obtain the best classification accuracy, four different sparseness levels (0, meaning no constraint; 0.75; 0.85; and 0.95) were tested. It was found that 0.85 sparseness gives the best results (Figure 3.2.3). The same sparseness level was found by Hoyer [2004] as the best level to learn Gabor-like filters from natural images. Hoyer [2004] defines the sparseness level as the relation of the L_1 norm to the L_2 norm. Where a sparseness of zero denotes the densest output vector, that is, when all outputs are equally active, and of one denotes the sparsest vector, when just one output is active. For equation and an illustration of different degrees of sparseness please see Hoyer [2004, pp. 1460-1461]. After NMFSC was trained on the train set, V was used (equations in section 2.2.1) to calculate the output on the occluded test set. For this, the obtained V was kept fixed and the optimization process was done for Y, approaching the predefined sparseness level for Y while trying to reduce the reconstruction error.

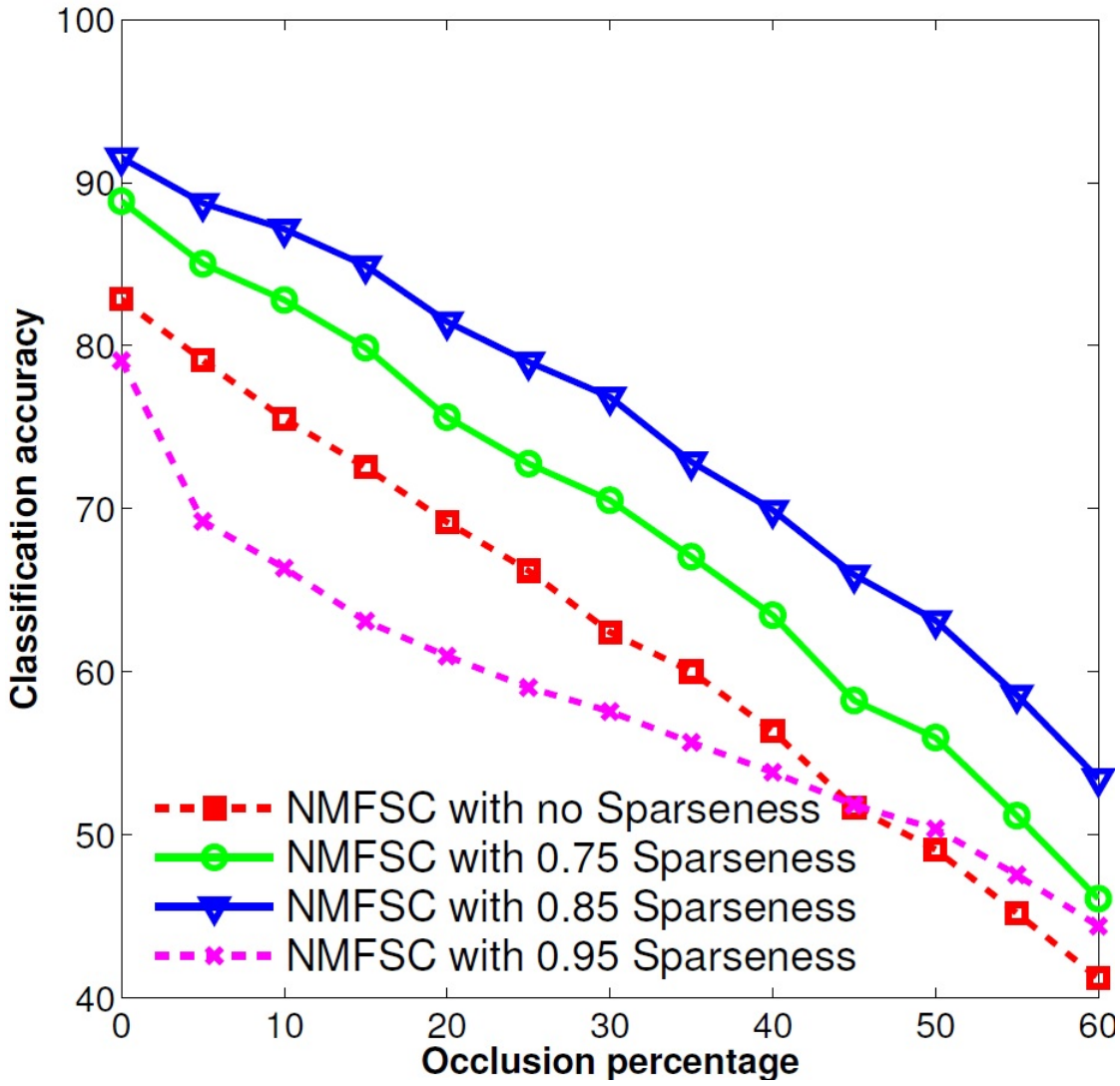


Figure 3.3: Effect of different sparseness levels on the robustness of NMFSC, using the occluded MNIST test set. A sparseness of 0.85 shows the best robustness. Very high or no sparseness reduces the performance.

3.2.4 Predictive coding/biased competition on MNIST

We trained PC/BC on 100,000 randomly chosen digits from the set of 60,000 images of the MNIST train set (obviously many digits were chosen more than once) and saved the weights for later calculation of outputs on the test set. After the learning was finished, each image of the test set was presented for 200 iterations to the network to achieve convergence of the outputs.

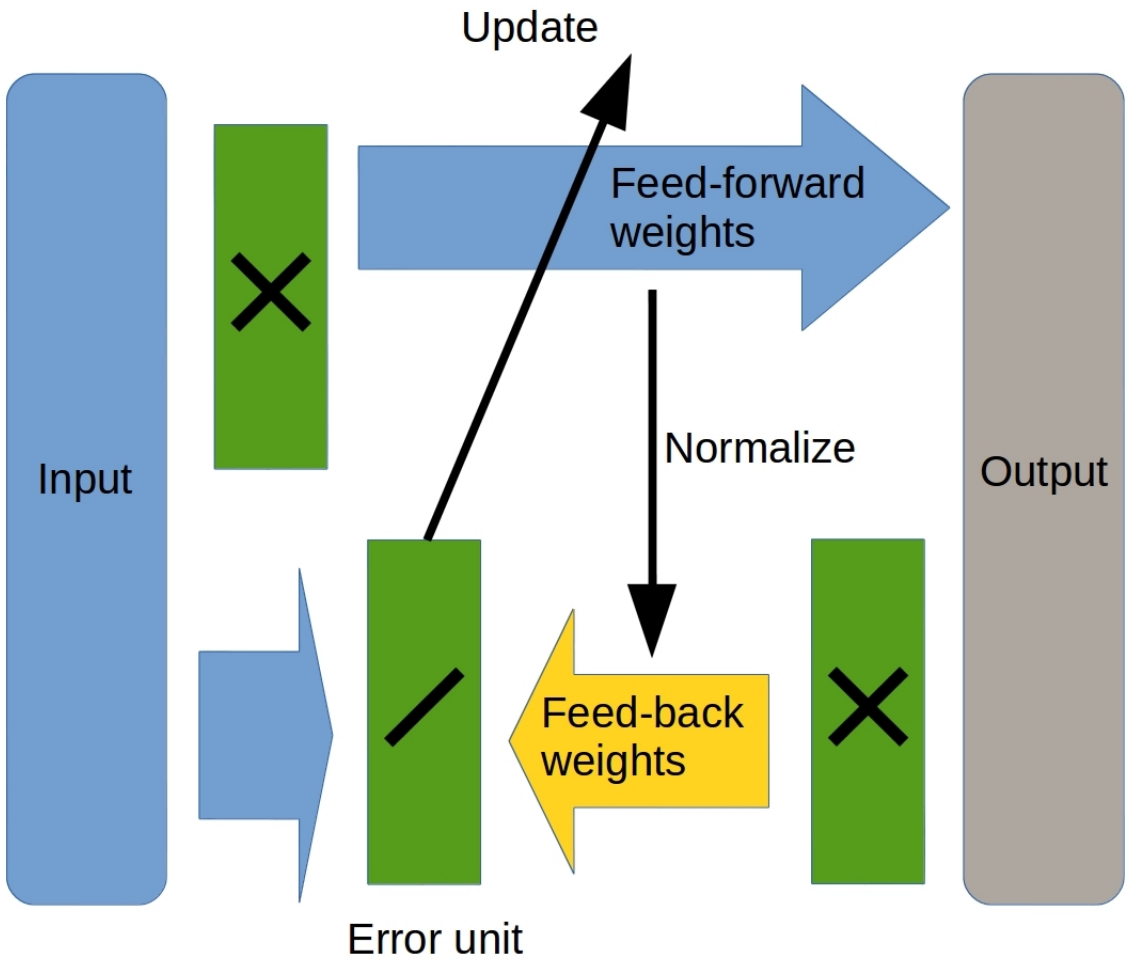


Figure 3.4: PC/BC diagram.

3.2.5 Classification

The accuracy of a classifier on the top of each method was considered as a criterion for robustness. The idea behind is that the classifier would face difficulties and decrease in performance if some information about the input is lost. Thus, a method with a more robust representation should have less decrease in accuracy under increasing levels of occlusion. It was decided to use a simple linear classifier, as the classifier should not compensate for a poor performance of the algorithms. Further, it is assumed that also the neural processing in the brain should facilitate linear classification (DiCarlo and Cox [2007]).

To measure the accuracy of classification, Linear Discriminant Analysis (LDA) was used on the output of the models on the test set. The MATLAB (2013a) function classify() with default parameters (linear discriminant function) was used for this purpose. The classifier was trained using the outputs of the respective methods on the train set.

3.2.6 Visualization of weights and receptive fields

The weight matrices of all methods were visualized to get an insight into how the data is processed. If the methods share a similar character in the extracted features or "weight organization", it can be assumed that this feed-forward part of the processing shares similarities. By looking that the receptive fields achieved form different methods, one could conclude that the competition is typically not changing their overall shapes, so the difference in performance should come from the difference in the functionality of the different competition mechanisms.

Hence, two approaches for visualizing the receptive fields were used. One was representing the weight matrix of a unit as gray-scale images. As the weight matrices correspond to the on-center and off-center inputs, these two parts were subtracted from each other (Wiltschut and Hamker [2009]). The strength of each of these weights was shown as the intensity of a pixel in the image, where, white denotes the maximum weight, gray denotes zero, and black the minimum weight. As an alternative, to visualize the receptive fields, reverse correlation was used. In order to obtain the optimal stimulus of a unit, images containing 90 random dots in front of black (zero) background were weighted with their corresponding outputs from a single unit. The average of the result was shown as the receptive field. This way, one can observe to which input parts each unit is sensitive. In other words, the resulting matrices visualize the correlation between the input and output values of each unit.

3.3 Results

3.3.1 Learned receptive fields

In order to verify if the models represent the input data in a comparable way, the weight vectors and receptive fields of 100 units for each model were visualized (cf. Section 3.2.6). To visualize the weight vectors of the Hebbian neural network (HNN), the feed-forward weight matrices were used (Figure 3.5a). For FastICA, the mixing matrix V was visualized (Figure 3.5c). The V matrix of basis vectors is visualized for NMFSC (Figure 3.5e). In PC/BC, the feed-forward matrices are shown (Figure 3.5g). For each method also the receptive fields estimated by reverse correlation are shown (Figure 3.5b,d,f,h), being not much different from the visualization of the weight matrices. All methods develop receptive fields with holistic forms of digits. Indeed, in NMFSC not all units show digit like shapes which may result from the chosen level of sparseness as mentioned in the methods.

3.3.2 Classification accuracy under occlusion

To investigate the differences in robustness to increasing levels of occlusions in the input, the classification accuracy of all methods as well as the raw data on the test set is measured. The experiments were rpeated 10 times with each algorithm under different starting conditions, i.e. randomly initialized weights. We do not show the error bars as they are zero for FastICA and NMFSC as they are deterministic and have been low for PC/BC and the HNN. It was observed (Figure 3.6) that FastICA does not improve the classification accuracy to that of the raw data. NMFSC causes a super-linear decrease of classification accuracy with respect to the linear increase of occlusion. PC/BC shows the highest robustness against occlusion. The robustness of the HNN is higher than NMFSC and lower than PC/BC. The methods having more "advanced" competitive mechanisms perform better under increasing occlusions.

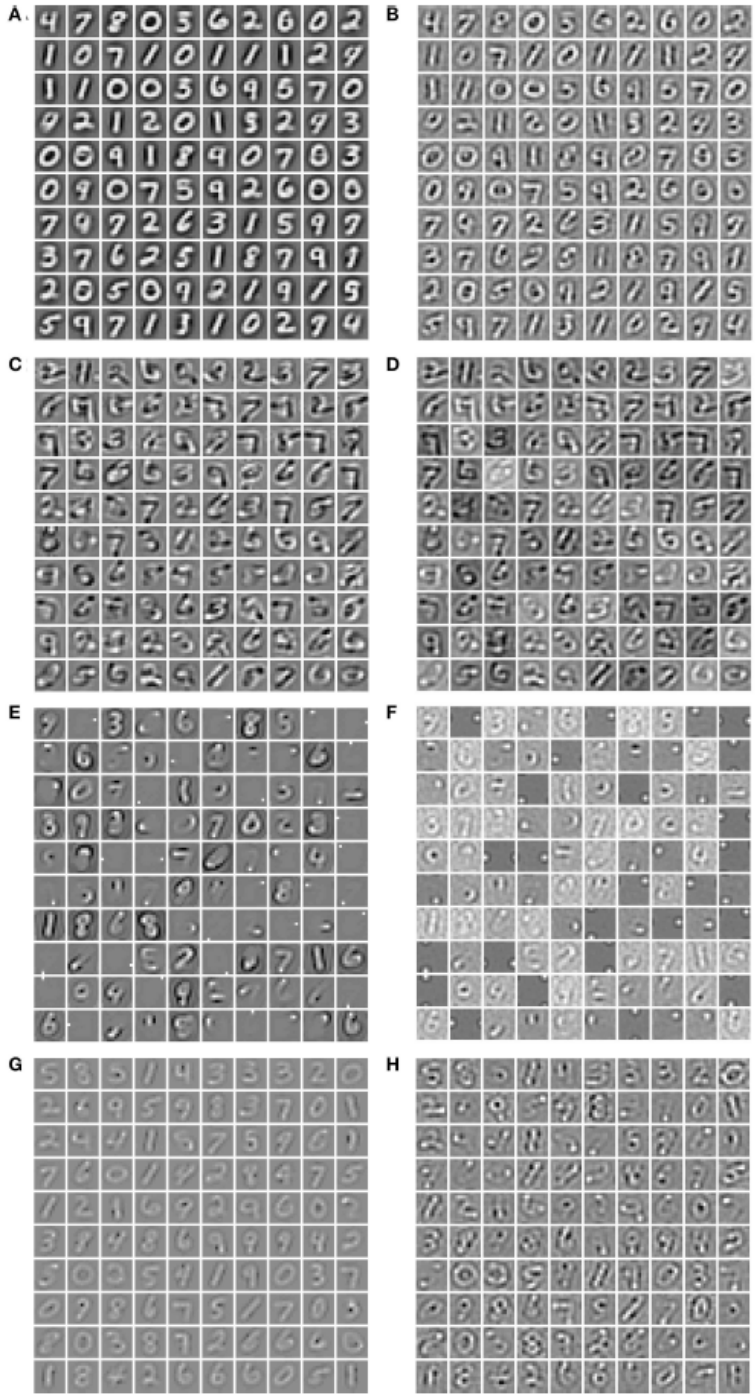


Figure 3.5: Visualization of the feed-forward weights and the receptive fields of 100 units, after training. Off-weights where subtracted from on-weights and each plot is scaled so that white denotes the maximum value and black the minimum. (a) The feed-forward weight matrices of the HNN and (b) its reverse correlation. (c) The component matrices of FastICA and (d) its reverse correlation. (e) The component matrices of NMFSC and (f) its reverse correlation. (g) The feed-forward weights of PC/BC and (h) its reverse correlation.

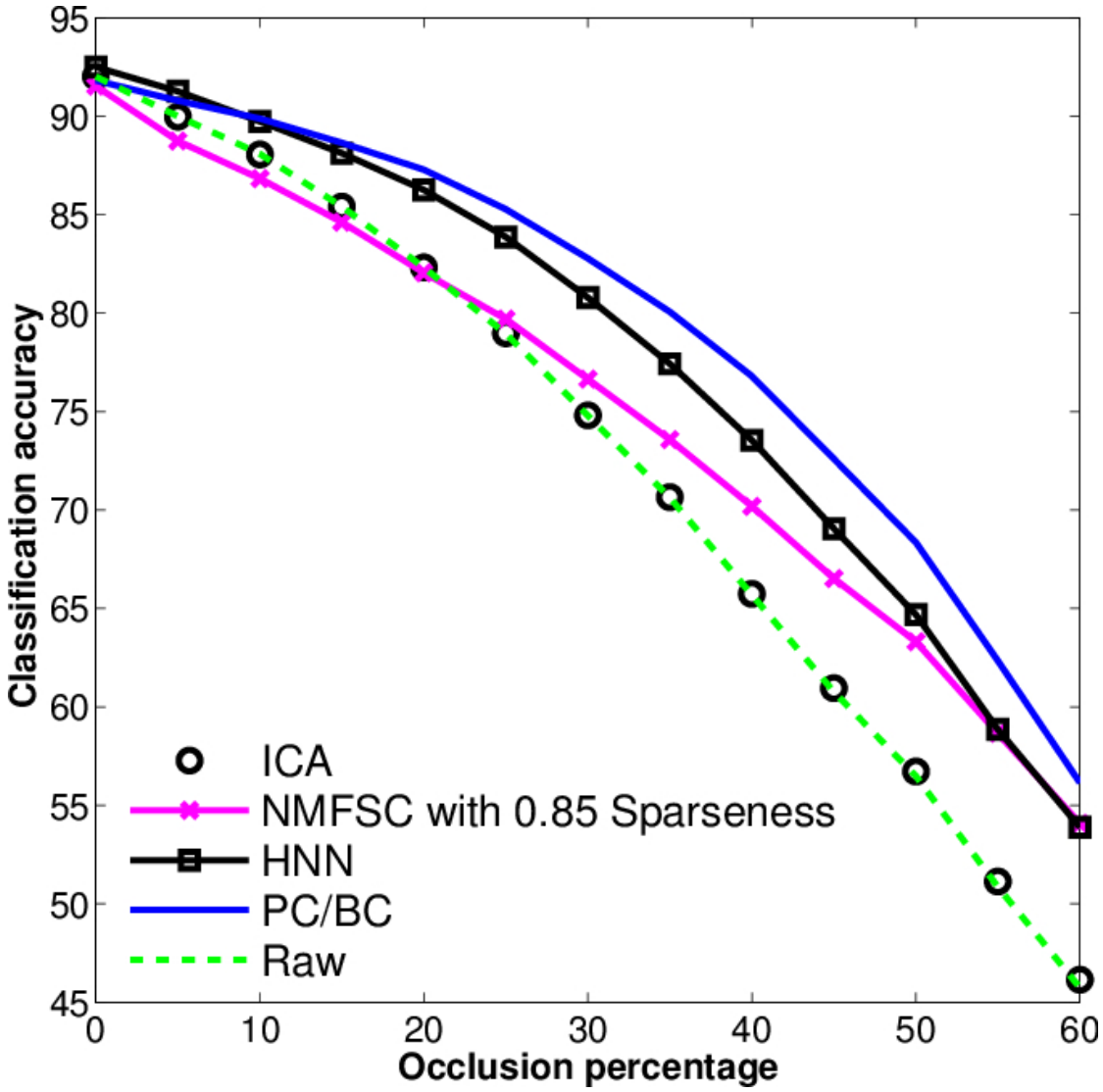


Figure 3.6: Classification accuracy on the output of FastICA, NMFSC, HNN, and PC/BC, using the occluded MNIST test set. Methods using competitive mechanisms show better robustness.

To further investigate the influence of the competitive mechanisms they are turned off for PC/BC and the HNN. That is, the lateral inhibitory connections were set to zero for the HNN, and only the first iteration step of PC/BC was used. The training of the classifier is repeated for these modified models. Both HNN and PC/BC cause a very low performance, even worse than the classification performance on the raw data when their competitive mechanisms are not used (Figure 3.7). This means that the competitive mechanism has a substantial influence on the accuracy under occlusion and the pure feed-forward processing is not enough to achieve robust recognition results.

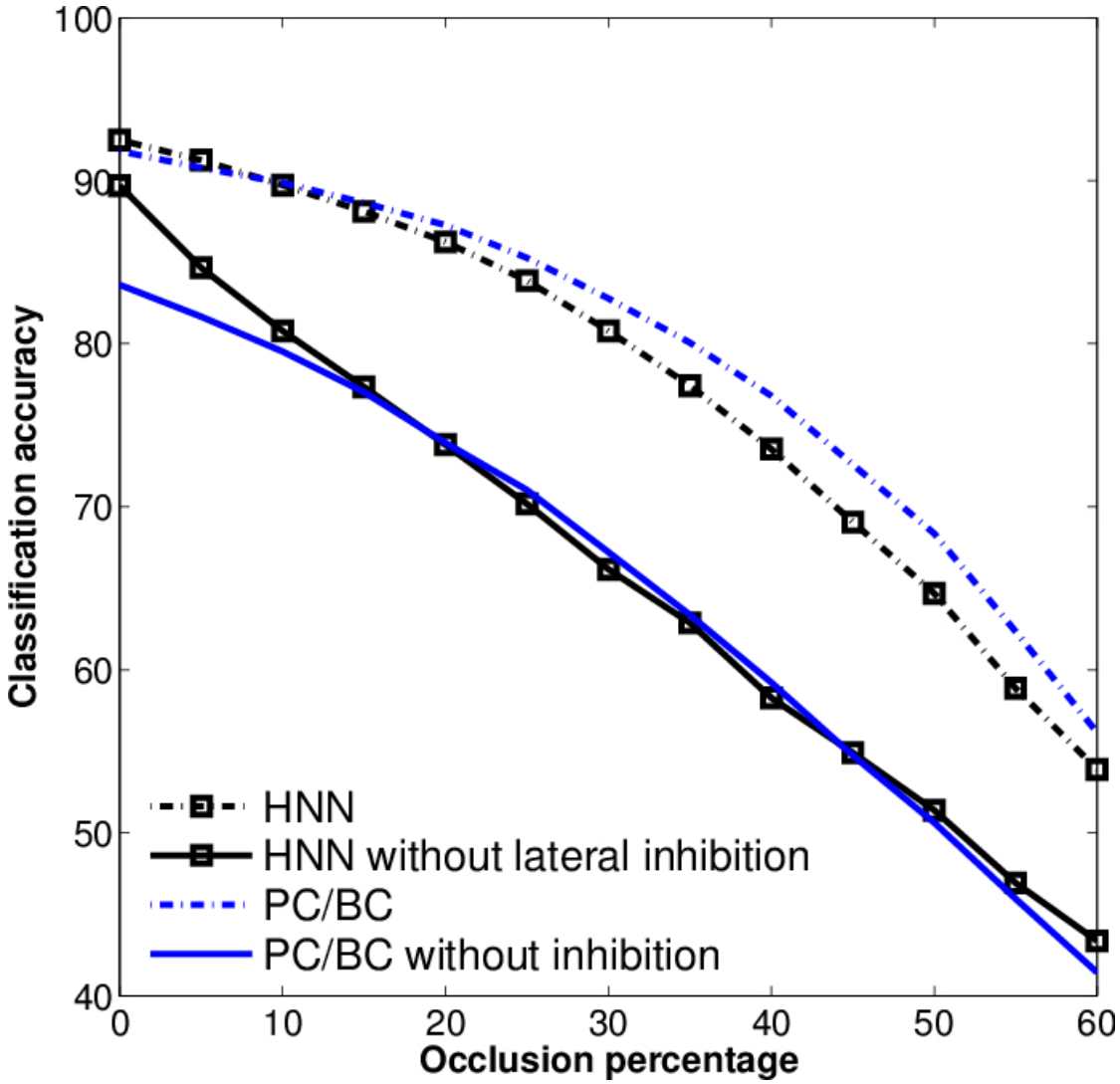


Figure 7. Robustness of the HNN and PC/BC with and without inhibition, using the occluded MNIST test set. Without the inhibitory connections the models show a sharp drop in performance against loss of information (occlusion).

3.3.3 Effect of occlusion on activity pattern

It is obvious that the activity pattern as a function of the input changes by increasing the occlusion in the input. The question is how stable the activity patterns of a method are when the occlusion in the input is increased. This is basically the same question as how much the classification accuracy is robust under loss of information. In Figures 3.8 to 3.11 the activity patterns corresponding to three random inputs under 0, 20, and 40 percent occlusion are illustrated. As one can see in NMFSC, HNN, and PC/BC the

Table 3.1: Cosine between non-occluded and occluded activity patterns, calculated on the test set with having particular occlusion levels. A cosine of 1 denotes an equal direction and 0 denotes an orthogonal one. The stability of the activity patterns confirm the results for the recognition accuracy, except the HNN shows a higher stability.

	20% occlusion	40% occlusion
FastICA	0.65	0.46
NMFSC	0.71	0.61
PC/BC	0.78	0.61
HNN	0.87	0.76

activity patterns corresponding to the non-occluded input and low occluded (20 percent) are comparable. In FastICA, though, the activity patterns are not easily comparable as ICA by nature produces very dense activity patterns. The activity pattern of FastICA on the (non-occluded) train set have a mean sparseness (Hoyer [2004]) of 0.41, which is, in comparison with NMFSC with 0.89, HNN with 0.80, and PC/BC with 0.89 sparseness, quite dense. However, in all methods, the activity pattern loses its original form when occlusion is increased.

To measure how stable the activity patterns of a method are under the four different levels of occlusion, the cosine of the angle between the non-occluded and the occluded activity vector was used. We calculated the cosine on the test set with 20% and 40% occlusion (Table 3.1) and found that methods showing more robust recognition accuracy also have less turn in their activity vector. Exceptionally, the HNN shows a more stable code than PC/BC based on this measure.

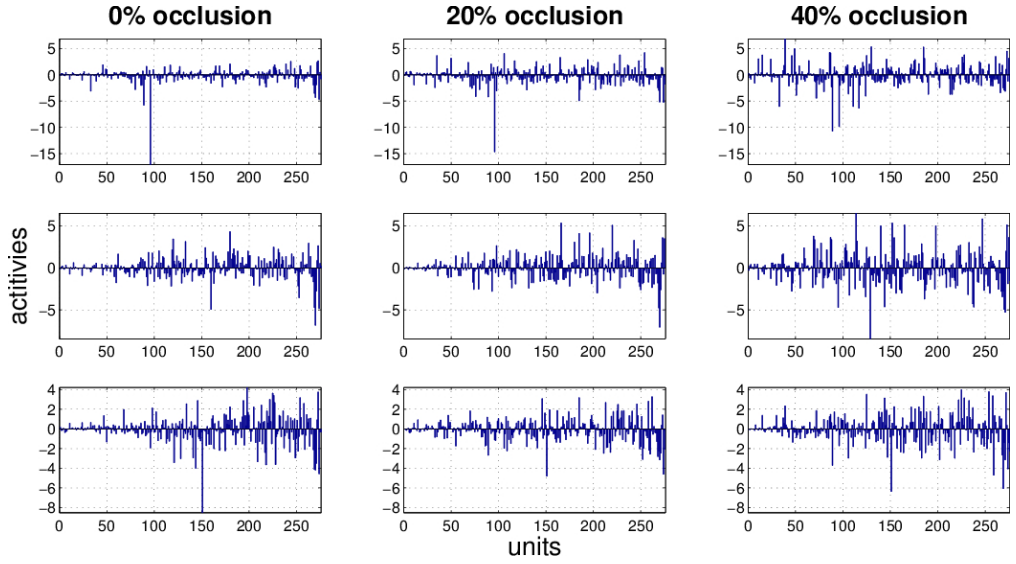


Figure 8. Three examples (row) how the activity pattern vary, under 0, 20, and 40 percent of occlusion (column) in FastICA.

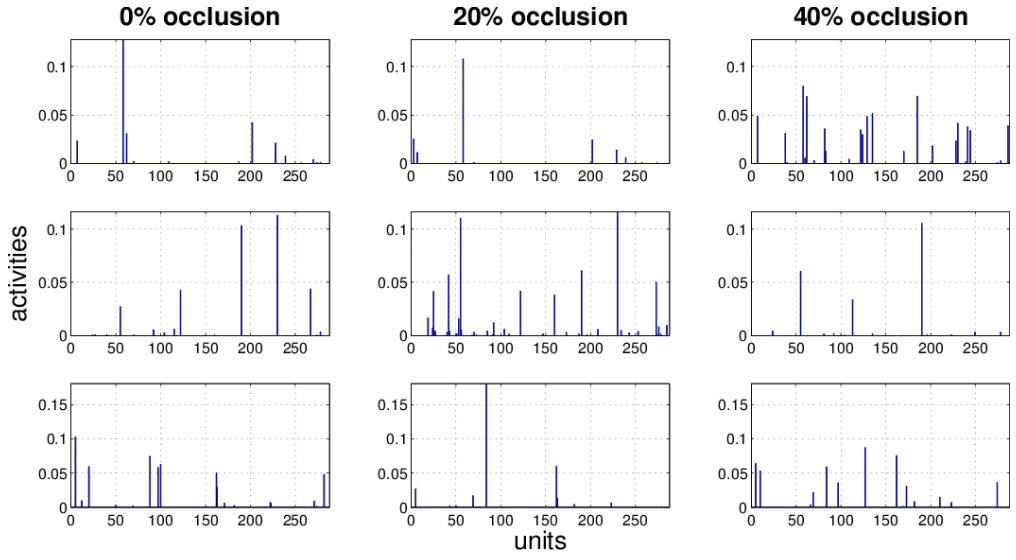


Figure 9. Three examples (row) how the activity pattern varies, under 0, 20, and 40 percent of occlusion (column) in NMFSC.

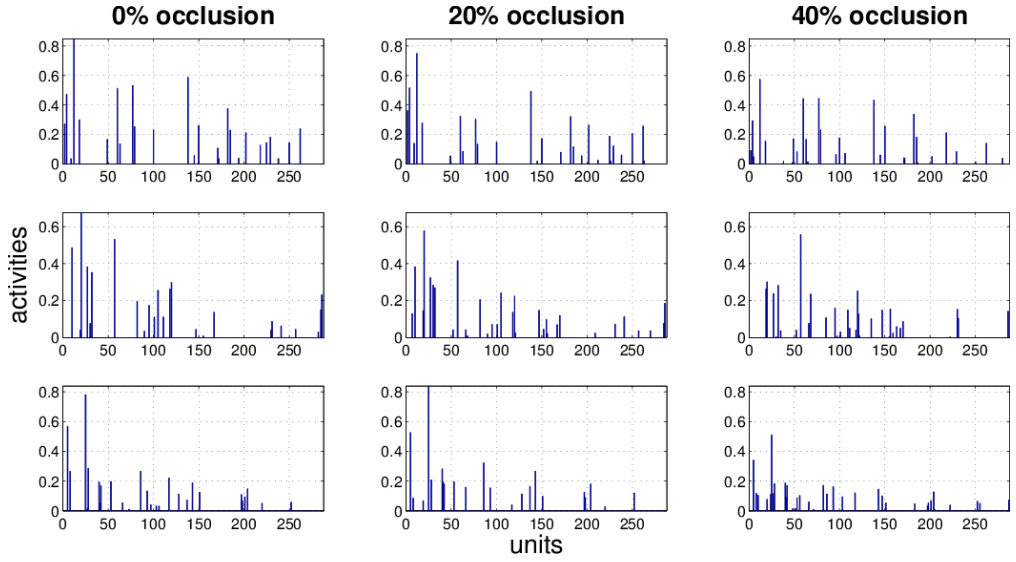


Figure 10. Three examples (row) how the activity pattern varies, under 0, 20, and 40 percent of occlusion (column) in HNN.

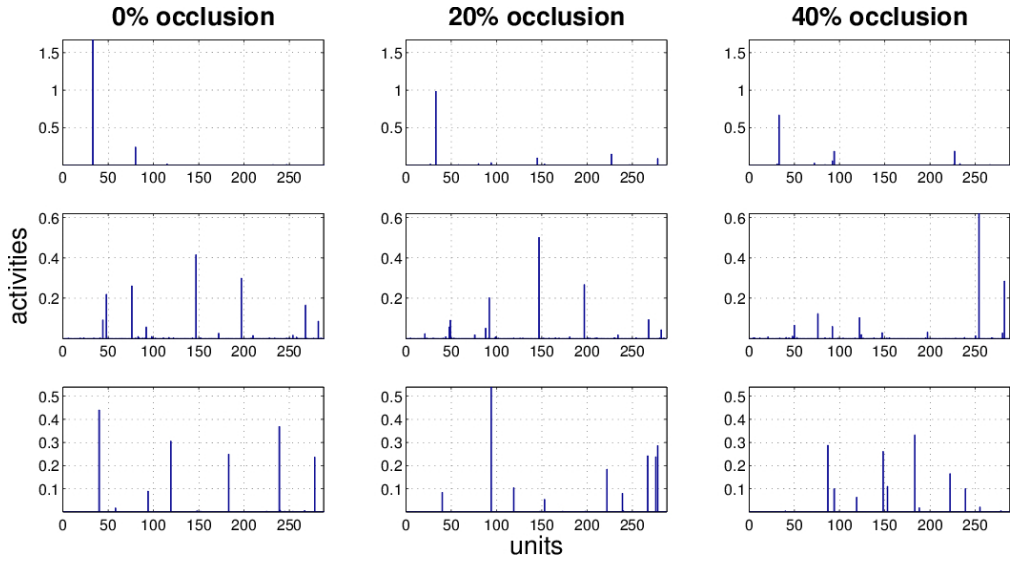


Figure 11. Three examples (row) how the activity pattern varies, under 0, 20, and 40 percent of occlusion (column) in PC/BC.

3.3.4 Selective inhibition in the Hebbian neural network

To investigate the selectivity of inhibition in the HNN, the relation between the strength of the lateral connections and the similarity of the feed-forward weights of a neuron to its laterally connected neurons was studied by visualizing the feed-forward weights of the

laterally connected neurons sorted by the strength of the outgoing lateral connections. Therefore, 10 neurons (left side) randomly selected and the weights of the laterally connected neuron were plotted (Figure 3.2.3). As one can see, the shape of the feed-forward weights of neurons being strongly inhibited are more similar to the weights of the inhibiting neuron than the ones which are less inhibited. That is, neurons have the strongest inhibition to neurons representing similar digits, mostly from the same class, followed by other classes sharing many similarities. This result was expected as the strength of the inhibition is relative to the correlation of the neurons.

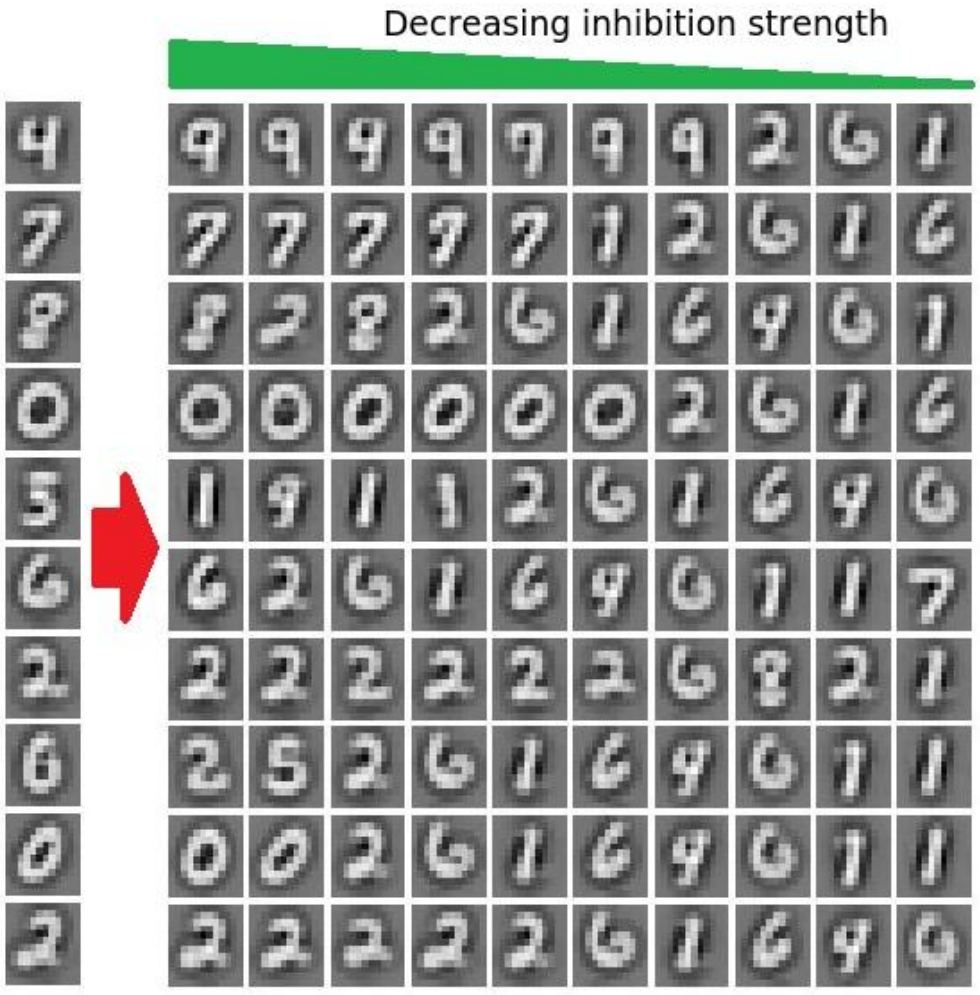


Figure 12. Selective inhibition in the HNN. On the left side the feed-forward weights of 10 randomly chosen neurons are illustrated. Right of each neuron, the weights of 10 neurons receiving inhibition from this neuron are plotted, sorted from left to right by descending lateral weight strength (inhibition). The illustration shows that neurons having more similar feed-forward weights are more inhibited than neurons having less similar weights.

3.4 Discussion

It was observed that the competitive mechanisms in the considered methods, FastICA, NMFSC, PC/BC, and HNN, have a direct effect on their robustness under loss of information. Results showed that all methods have developed receptive fields similar to digit shapes and thus the methods should be comparable. Apparently, this similarity itself cannot be used as a criterion for robustness against loss of information (occlusion). It was observed that the receptive fields of FastICA are more similar to digits than those of NMFSC, although, NMFSC shows a better accuracy under occlusion. However, without using its competition mechanism it behaves worse than FastICA. Further, HNN and PC/BC have the clearest receptive fields and the highest performances, indeed, without the competitive mechanism their accuracy drops lower than FastICA and NMFSC. Also, the recognition accuracies of PC/BC and HNN with and without competition cannot be explained by differences in the receptive field shapes. Without competition, HNN behaves slightly better than PC/BC, whereas with competition PC/BC shows better accuracy. This means the receptive field quality alone does not cause the observed higher robustness. Without occlusions no method showed a strong superiority in the accuracy, indeed, they showed clear differences when the input was distorted. Some models are more stable when the input is occluded. This stability is in line with the results of the classification accuracy. While the HNN shows the least change in the cosine between its population responses with and without occlusion, its classification accuracy is a bit weaker than that of PC/BC for larger occlusions. The two dominant methods in this study, the HNN and PC/BC, employ different mechanisms for competition. These mechanisms help the systems to selectively inhibit the outputs of other units or respectively their input. To observe to which extent competition enhances robustness under occlusion, the classification performance when the competitive mechanisms were turned off was evaluated. When the mechanisms are off, PC/BC and the HNN show a very low performance in the robustness to occlusions, as NMFSC without using the sparseness constraint. So obviously, the feed-forward processing is not enough to obtain a sufficiently differentiated output and it can be assumed that competition is playing an essential role in the robustness of these systems. It was also observed that methods benefiting from a competitive mechanism are superior to FastICA having no

competitive mechanism in computing the output. FastICA linearly transfers the input space into a new space with least dependent components. When facing an image, FastICA produces a dense set of activities to describe the image in the new space. NMFSC without sparseness constraint performs like FastICA. However, when a reasonable level of sparseness is set for the activities of NMFSC it outperforms FastICA. The reason is that the sparseness constraint prevents the appearance of redundant information to some extent. Indeed, a too sparse representation can remove some useful information and result in reduced accuracy. However, NMFSC acts weaker than the Hebbian neural network and PC/BC which may depend on the subtractive updating rule for output competition. Moreover, the optimal sparseness level is practically impossible, since a priori knowledge about the number features for an optimal representation is needed (Spratling [2006]). Also having this knowledge does not have to lead to an optimal result as different classes often need different numbers of active features. Among the three generative models, FastICA, NMFSC, and PC/BC, PC/BC has been the superior model in this experiment. It uses a multiplicative updating rule to calculate the output activity. It finds the best matching units and removes their representations from the input of the other units, producing a sparse output while approaching a minimal reconstruction error. This "online" error minimization is realized by iteratively updating the error units representing the local elements of the reconstruction error and driving the output units. The HNN also has a competitive mechanism according to which the best matching units suppress other ones. In contrast to PC/BC, which tries to minimize the reconstruction error, the Hebbian Neural Network, as a whole, does not approach any explicit objective. It only exploits the knowledge about the co-activity of units in order to suppress them. Thus, stronger units suppress potentially confusing weaker ones. That is, in HNN each unit is competing with other units based on its learned, local inhibitory weights, whereas PC/BC is actively using its distributed representation of the reconstruction error to minimize a global error signal. This additional information PC/BC uses may be the reason for the slight advantage of PC/BC against HNN for larger occlusions. The conclusion is that in order to achieve high robustness against loss of information in object recognition one should focus on improving the competitive mechanism. Competition between units seems to play a major role in preventing the system from producing redundant activities. The experiments also give evidence that the cortical mechanisms of competition, as

lateral inhibition, are the source of its robust recognition performance, even on a single layer level. Similar effects to our V1 based evaluation can be found in deeper models of the visual cortex ventral stream, where even inhibitory lateral connections play an important role in robustness to occlusions (O'Reilly et al. [2013]).

Chapter 4

Excitation/Inhibition balance and dis-inhibition

Simulating the inhibitory mechanism is a challenge which requires answering several questions. Two of the most important ones are tried to be answered here. The **first question** is, how much inhibition is useful and when it is necessary. Weak inhibition could cause redundant activities and correlation among units. Too strong inhibition, however, would not allow units to be active when they should be. The **second question** is whether inhibition should exist everywhere all the times.

Excitation/Inhibition balance

The **first question** studies the proportion of excitatory signal a neuron receives to the inhibitory signal approaching that neuron. There exist several known advantages of the balanced excitation/inhibition of which one could count: stabilization (Shadlen and Newsome [1998], Amit and Brunel [1997], Litwin-Kumar et al. [2016], Kelly [1990], Arkachar and Wagh [2007]) , gain control (van Versendaal and Levelt [2016], Chance et al. [2002], Wehr and Zador [2003]) and dynamic range of responses (Shadlen and Newsome [1998], Burkitt [2001]). Here the attempt is to make a balance between excitation and inhibition by a modification in the anti-Hebbian learning. The proposed dynamic weight reduction method for inhibitory weights guarantees that the inhibitory weights appear when they are necessary and that they are reduced when they could cause loss of information. This concept is explained in section 4.1.3.

Winner-take-all and dis-inhibition : The **second question** talks about the exceptional removal of inhibition applied to some neurons and letting them be the absolute winners of the competition. Although the excitation/inhibition balance has been under attention for a while, local dis-inhibition has recently come to attention (Karnani et al. [2016]). Karnani et al. [2016] unveils that the appearance of some holes in the "blanket" of inhibition which covers all neurons all times, lets some neurons in the brain to freely fire without being suppressed by inhibitory neurons.

The inhibitory cells are very diverse in their characteristics, but they are mainly grouped into three subsets; Protein Parvalbumin (PV), Neuropeptides Somatostatin (SOM) and Vasoactive Intestinal Polypeptide (VIP) (Rudy et al. [2011]). PV cells play a significant role in controlling the timing which is out of the scope of this thesis. SOM cells are responsible for inhibiting the pyramidal cells and implying competition among them. In the end, VIP cells do the interesting task of inhibiting the SOM inhibitory cells in small areas close to their pre-synaptic connections from pyramidal cells (Karnani et al. [2016]). This way, the VIP cells dis-inhibit the excitatory cells and let them freely fire and thus inducing a Winner-Take-All (WTA) mechanism among neurons. Here WTA means the mechanism which favors the winner of the competition in the sense that the winner among some neurons gains the advantage of some function like the permission to fire (Lemmon and Vijaya Kumar [1989]). Although any competition could have a winner, it does not mean that the winner is not inhibited by the losers. As mentioned in Karnani et al. [2016], the pyramidal excitatory cells are under a blanket of inhibition which covers all neurons including the winners. The new biological data suggest that this inhibition may not be necessary in all places all times; sometimes a hole may appear in the blanket of inhibition. There is no absolute explanation of this phenomenon, but experiments with dis-inhibition introduced in this thesis may be a start in explaining why and where the dis-inhibition may be necessary. The mathematical model suggested here does not have the exact architecture of the inhibitory mechanism in the brain, yet it could explain this system in a simplified manner.

4.1 The model

4.1.1 The neural network

The neural network consists of two layers of rate based neurons. The learning in the feed-forward neurons is based on a modified Hebbian learning method (Teichmann et al. [2012], Kermani Kolankeh et al. [2015]) which controls the weight growth based on the Oja's rule but the strength of the reduction is adaptive based on the activity of the neurons. The formula for updating the feed-forward weights is as follows:

$$\begin{aligned}
\tau_H \frac{\partial H}{\partial t} &= ((r^{post} - \gamma)^+)^2 - K - H \\
\tau_\alpha \frac{\partial \alpha}{\partial t} &= -\alpha_{decay} + H + ((net^{post} - 2)^+)^2 \\
\tau_{Learn} &= base_a + base_b \cdot \exp(-base_c \cdot (Ca^{post})^+) \\
\tau_{Learn} \frac{\partial w}{\partial t} &= (r^{pre} - \tilde{r}^{pre}) \cdot \left(Ca^{post}\right) - \alpha \cdot \left(Ca^{post}\right)^2 \cdot w
\end{aligned} \tag{4.1}$$

where K = 0.05, γ = 0.7, τ_H = 100, α_{decay} = 0.0005, τ_α = 10000, $base_a$ = 5000, $base_b$ = 30000, $base_c$ = 15, net^{post} is the "net" activity (dot production of the input to the weight matrix) of the post synaptic neuron and

$$x^+ = \begin{cases} x & x > 0.0 \\ 0 & else \end{cases} \tag{4.2}$$

The activity of a neurons is a function of the membrane potential which in turn is the excitation a neuron receives form the synapses subtracted by the inhibition it receives from the neighboring neurons. The membrane potential is calculated as follows:

$$\tau_m \frac{\partial m_j}{\partial t} = -m_j + \sum_i w_{ij} r_i^{Input} - \sum_{k,k \neq j} c_{kj} r_k \tag{4.3}$$

with $\tau_m = 10ms$.

At the end, the transfer function:

$$r_j = \begin{cases} 0.5 + \frac{1}{1+e^{-3.5(m_j-1)}} & \text{if } m_j > 1 \\ (m_j)^+ & \text{else} \end{cases} \tag{4.4}$$

turns the membrane potential to an activity which could not be higher than 1.5.

4.1.2 Adaptive α

The weight reduction factor, α, in the feed-forward weights is dynamic and not only prevents the weights from unlimited growth but also induces sparsity to the population code. α changes based on the activity of the neuron. As it is shown in formula 4.1, α follows the square of the overhead of **net activity** compared to 2 plus the value of another parameter, H. H, in turn, follows the overhead of the **activity** of the network compared to $\gamma = 0.7$. The idea is that if the **net activity** of the neuron is high, greater than 2, for a long period, and/or the **activity** of the neuron is higher than γ in a much longer period, then the weight reduction factor should be increased to reduce the weights with a higher speed. Time constants $\tau_H = 100$ and $\tau_\alpha = 10000$ are responsible for regulating the speed with which the parameters are changed.

4.1.3 Inhibition

Inhibition was applied to a neuron by subtracting from the actual neuron the dot product of the activities of the other neurons in a limited neighborhood to the inhibitory weights connecting those neurons to the actual neuron:
[net activity of the actual neuron] $-$ [activities of other neurons in the neighborhood] $\cdot$ [inhibitory weights arriving to the actual neuron from the neighboring neurons]
This is shown also in equation 4.3. Further, in this section, several learning methods for learning the inhibitory weights are explained. All these methods are implemented and their performances are compared. The Dynamic Inhibition (DI) explained in 4.1.3 and the Dis-Inhibition introduced in 4.1.3 are inventions of the author and the rest of the inhibitory mechanisms: Static Inhibition 4.1.3, Foldiak's Inhibition method, 4.1.3 and King's Inhibition method 4.1.3 are used for comparison. One should note that the

methods from other authors are applied just for learning the inhibitory weights, and the feed-forward weights use a common learning rule.

Static Inhibition

The static mechanism is the same as in Teichmann et al. [2012] and Kermani Kolankeh et al. [2015].

$$\tau_{AH}\frac{\partial w}{\partial t} = r^{pre}\cdot(r^{post}-\theta)^{+} - \alpha_{AH}\cdot r^{post}\cdot w \tag{4.5}$$

where τ_{AH}=10000 for the first layer and τ_{AH}=50000 for the second layer, $\alpha_{AH} = 0.3$ and $\theta = 0.05$. In the static learning rule for the inhibitory weights, the weights are increased as a function of the correlation of the pre- and post-synaptic activities excluding the cases the post-synaptic activity is too low (lower than θ). The weights are reduced based on a portion of the multiplication of the pre-synaptic activity multiplied by the weight. The idea is that if with low pre-synaptic activities, high post-synaptic activities are produced, the conclusion would be that the weights are too big and should be reduced.

Foldiak's Inhibition method

This method is an implementation of the Foldiak's method for learning inhibitory weights. This version of Foldiak's method is equivalent to the one introduced in Foldiak [1990], but easier to implement (King et al. [2013]):

$$\tau_{AH}\frac{\partial w}{\partial t} = r^{pre}\cdot r^{post} - {r_{\Sigma}}^{pre}\cdot {r_{\Sigma}}^{post} \tag{4.6}$$

where r_{Σ} is the temporal mean of r with the time constant $\tau_{sm} = 500$.

$$\tau_{sm}\frac{\partial r_{\Sigma}}{\partial t} = r - r_{\Sigma}$$

In this method, constant simultaneous high values of the pre- and post-synaptic activities are considered as the sign of existence of some too strong weights and make the negative part of the equation 4.6 stronger, causing to weight reduction.

King's Inhibition method

This relatively recent method was introduced by King et al. [2013].

$$\tau_{AH}\frac{\partial w}{\partial t} = r^{pre} \cdot r^{post} - {r_\Sigma}^{pre} \cdot {r_\Sigma}^{post} \cdot (1 + w) \tag{4.7}$$

$$\tau_{sm}\frac{\partial r_\Sigma}{\partial t} = r - r_\Sigma$$

Here basically the idea is the same as in Foldiak's method, with the difference that in the Foldiak's method the weight reduction was proportional to the weight. The $+1$ is added to ensure the reduction even if the weights are too small. This means, in average this reduction is stronger than what was used in Foldiak's methods. Also, if there is an inhibitory weight but already there is no or low correlation, the $(1 + w)$ factor causes the reduction of the inhibitory weight.

Dynamic Inhibition

The same as in the static way(Teichmann et al. [2012] and Kermani Kolankeh et al. [2015]), in the dynamic method also the weights are increased based on the correlation between the pre- and post-synaptic activities. The equation for learning and dynamically decreasing the lateral weights is as follows:

$$\tau_{AH}\frac{\partial w}{\partial t} = (r^{pre} \cdot (r^{post} - \theta)^+)^2 - \alpha \tag{4.8}$$

where

$$\alpha = (net^{post} - mp^{post})^2/\eta$$

with τ_{AH}=10000, $\theta = 0.05$, $\eta = 100$ for the first layer and $\eta = 150$ for the second layer.

The static method follows a rule, based on the Oja's weight reduction method, but the dynamic method decreases the lateral inhibitory weights more intuitively. Oja proves that his subtractive weight reduction process is equivalent to a divisive normalization which normalizes the weight vectors to have the unit length. In contrary, the dynamic weight reduction method decreases the weights based on the difference between the weighted sum of the inputs and the membrane potential. The idea is that if the inhibition has been applied to the neuron for a long time but yet the neuron's weight vector is not turned to a direction which causes less correlation with the other neurons, the inhibitory weights to the neuron should be reduced.

In the condition when we need some slightly overlapping features to describe the input, obviously they will have some unavoidable correlation. In this situation, reducing the inhibitory signal in the case of unsuccessful inhibition would help the model to learn useful overlapping features. Here we talk about the overlap in the feature space. Besides, DI prevents strong inhibitory signals which may push the membrane potential to negative values and simply prevent the neurons from turning their weight vector to the correct direction.

Dis-inhibition on the second layer

One could explain this mechanism in short as follows: stop sending an inhibitory signal to the winner neuron by setting the inhibitory weight to that neuron to zero. In the invented model here, two groups of inhibitory weights are employed: one is the inhibitory weights from the inhibitory cells to the excitatory ones and the other one is the inhibitory weights from the inhibitory neurons to the inhibitory neurons. The dis-inhibition is considered for both types of inhibition.

What the dis-inhibition brings, is inducing a kind of winner takes all mechanism in which the winner takes the complete chance of learning and the loser is suppressed

based on the level of the correlation it has with the winner. The inhibitory weight on the second layer are learned based on the following rule:

$$\tau_{AH}\frac{\partial w}{\partial t} = (r^{pre} \cdot (r^{post} - \theta)^{+})^2 - \alpha - \beta \tag{4.9}$$

where τ_{AH}=10000, α is the same as for the dynamic inhibition and β is determined as follows:

$$\beta = \begin{cases} \infty & ca^{post} >= r^{pre} \\ 0.0 & else \end{cases} \tag{4.10}$$

In this rule, in the case the current neuron's activity, that is, the pre-synaptic activity is less than the activity of the neuron which is being inhibited by the current neuron, the inhibitory weight to that neuron is set to zero, declaring it the absolute winner of the competition. One should pay attention that this does not mean the permanent removal of the inhibitory weight because the weight increase is additive, not multiplicative. So a zero synaptic weight also could increase as the result of the correlation between the two ends (pre- and post-synaptic activities) of it.

4.1.4 Network architecture

The network consists of two layers of neurons. Comparable to what is usually seen in the deep learning models, in this model also each layer consists of different maps and the number of maps is higher in the second layer than in the first layer. By maps one means the number of neurons which simultaneously look at the same part of the input. Experimentally 4 maps were decided for the first layer and 16 maps for the second layer. In this thesis, the input is not considered as a layer, although the input actually consists of two maps for positive and negated negative parts of the whitened image. The number of excitatory neurons was chosen so that the whole input surface was covered. Obviously, the input to the second layer is the output of the first layer which should also

be completely covered. On both layers, the receptive fields were 5×5. Receptive fields were of 1-pixel distance from each-other, and no receptive field was partially out of the input surface. This means the network had $576 = (28 - 5 + 1)^2$ excitatory neurons on each of the 4 maps of the first layer and $400 = (24 - 5 + 1)^2$ neurons on each of the 16 maps of the second layer. In contrast to the convolutional neural networks, no weigh sharing was employed and the neurons did not share their weight matrices. Neurons in every 5×5 neighborhood are inhibited by an inhibitory neuron receiving excitatory weights both from the input and outputs of the neurons. Each inhibitory neuron receives inhibitory connections from a 5×5 neighborhood around it. Figure 4.1 could give a better impression how the neurons are connected to each other.

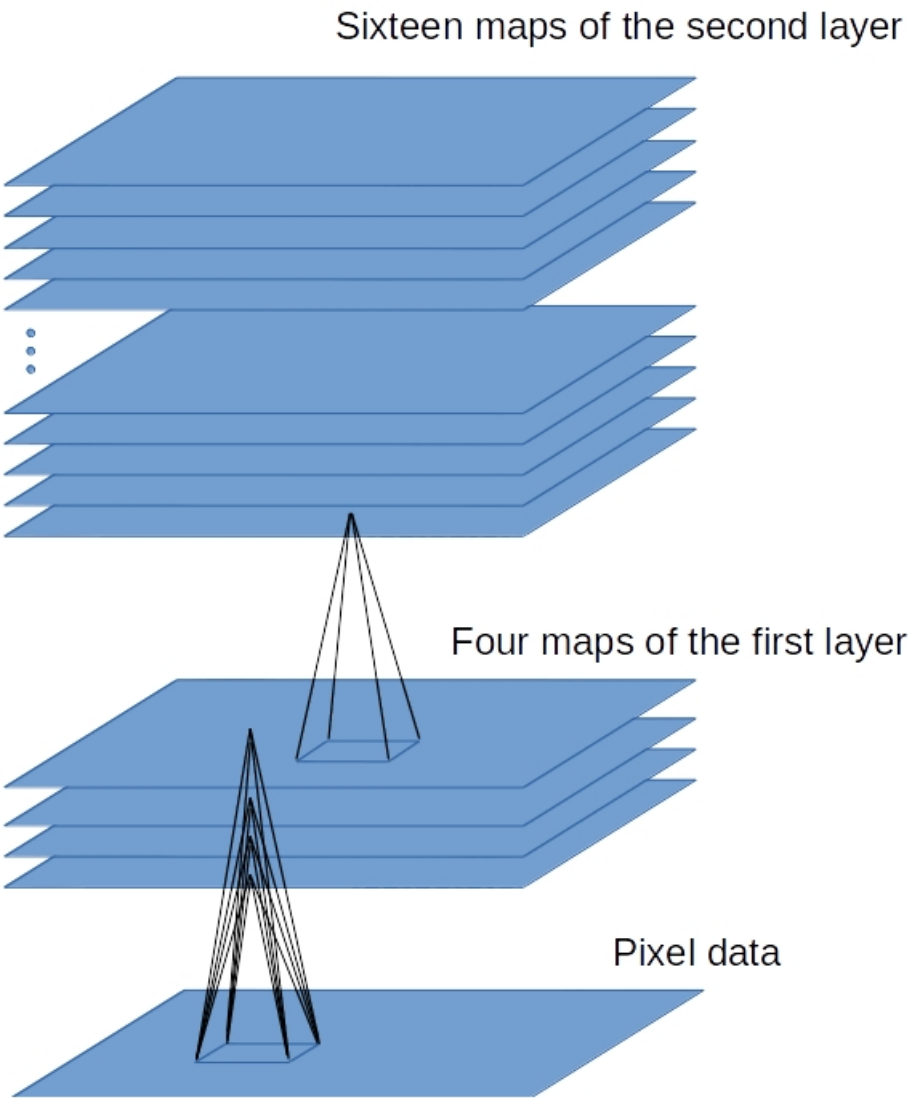


Figure 4.1: The schema of the network

4.2 Materials and methods

The mentioned methods in the previous section were applied to MNIST datasets of handwritten digit images.

4.2.1 Input preprocessing, learning and evaluation on MNIST

The preprocessing applied to the MNIST digits used in this part of the thesis are exactly the same as what was applied in Kermani Kolankeh et al. [2015] with the exception that the digits were not resized and the original 28*28 pixel images were used. Each image was whitened and the result of whitening of each image was a new image with positive and negative values. each whitened image was then separated into two images with the same size as the original image but one containing the positive and the other one the negative values. The image with the negative values was then negated to have positive values. This was done because of the biologically plausibility of the model; no negative input is possible.

Learning, Train and test phases

Like in the section 3, here also the study consists of 2 main phases: learning and evaluation of the network parameters. The evaluation itself has 2 phases; training and testing the **classifier** based on the activities of the network. In this section, they are simply called train and test phases. So, we have three phases: learning, train, and test. For learning and train, 60000 train samples and for testing 10000 samples from the train and test sets of MNIST dataset of handwritten digits were used. Each handwritten digit was considered as a separate image for preprocessing and learning.

As it is explained in section 4.1.4, the receptive fields are 5×5 pixels and each does not cover the whole input image. In order to increase the probability of visiting a part of a digit and also causing shift and rotation invariance, for the learning phase each digit was first randomly rotated between -30 and 30 degrees (with 10-degree steps) and then shifted randomly by 0 to 10 pixels to a random direction. Without the random shift and rotations, because MNIST digits are centralized, the synapses out of the central part of the receptive field would not visit any input value other than zero backgrounds. Besides, the shift and rotation used in the learning phase help the network to learn components which could bring shift and rotation invariance to the network. In the **learning phase**, each shifted input image was shown to the network for 100 milliseconds (steps) and in each step, the activities were updated. On the step 100th the weights were updated and then a new input was fed to the network for the next 100 steps and so on.

In the **train phase**, unlike in the learning phase, the original (not shifted but just whitened) digits were fed to the network and the activities were saved for further analysis. In the **test phase** based on the type of invariance under study, different kinds of manipulation were applied to the input. For investigating shift invariance, five different levels of shift were applied to the test set: as the results of 0, 1, 2, 3 or 4 pixels random shift to each digit in the test set of the MNIST dataset, five different test data sets were produced. Obviously the test set with zero pixel set is the same as the original test set. As for the rotation, as the result of 4 different levels of rotation, 4 different versions of the test set were produced and tested on the network. These four different test sets were the results of 0, 10, 20 or 30 degrees of rotation of each digit in a random direction. The noise invariance was examined with n-MNIST dataset which will be explained later in this chapter.

In train and test phases each digit image was shown to the network for 100 steps and on each step the activities were updated. On the step 100th, the activity was saved for classification. Obviously in the evaluation phase, the weights were not updated, but in the learning phase, at the end of each step 100 steps, the weights were updated.

4.2.2 Classification

In order to measure the performance of the network, the population activities of each layer were considered for the later classification. The classification was done by using the MATLAB "classify" function which applies a linear classification by fitting a multivariate normal density to each group of data, with a pooled estimate of covariance (https://www.mathworks.com/help/stats/classify.html).

4.2.3 Visualization of the second layer receptive fields

Unlike visualization of the cells of the first layer which is simply the illustration of the weight matrix of each cell, the visualization of receptive fields of higher layers is more complicated. The visualization used here is inspired from Zeiler and Fergus [2014] but a bit simplified, as in contrast with their work the current work does not have the challenge of pooling. Also, the complete "up-down" system they employ is not used

here. The visualization algorithm is explained step by step: first look for the highest activity of the upper layer neuron according to the whole input set of images; then find the corresponding activities of the neurons in the lower layer whose activities are the inputs to the upper layer neuron; next, multiply the receptive fields of the lower layer neurons to their activities; afterward multiply each of the resulting matrices to the weight which connects the corresponding receptive field to the higher layer neurons. This approach weights the lower layer receptive fields by their long-term influence on the upper layer neuron. Long-term influence means the strength of the feed-forward weight connecting them to the higher layer neuron. In the end, the mean of all weighted receptive fields was calculated and multiplied to (weighted with) the activity of the higher (second) layer neuron.

4.3 Results

Receptive field shape

As it is shown on figure 4.2 the neurons learn Gabor like receptive fields. This is in contrast to the previous work (Kermani Kolankeh et al. [2015]) in which digit templates were learned. The difference comes from the fact that in the current work the digits are randomly shifted and also the receptive fields are smaller than the complete input surface. Although different competition methods change the classification performance of the network, the receptive fields achieved from different methods are visually similar.

In figure 4.3 the receptive fields of the second layer are visualized which look more complex than Gabor filters. They are also bigger than the lower layer receptive fields. The Gabor like filters achieved on the first layer are 5×5, while the second layer receptive fields cover larger areas of the input with 9×9 filters.

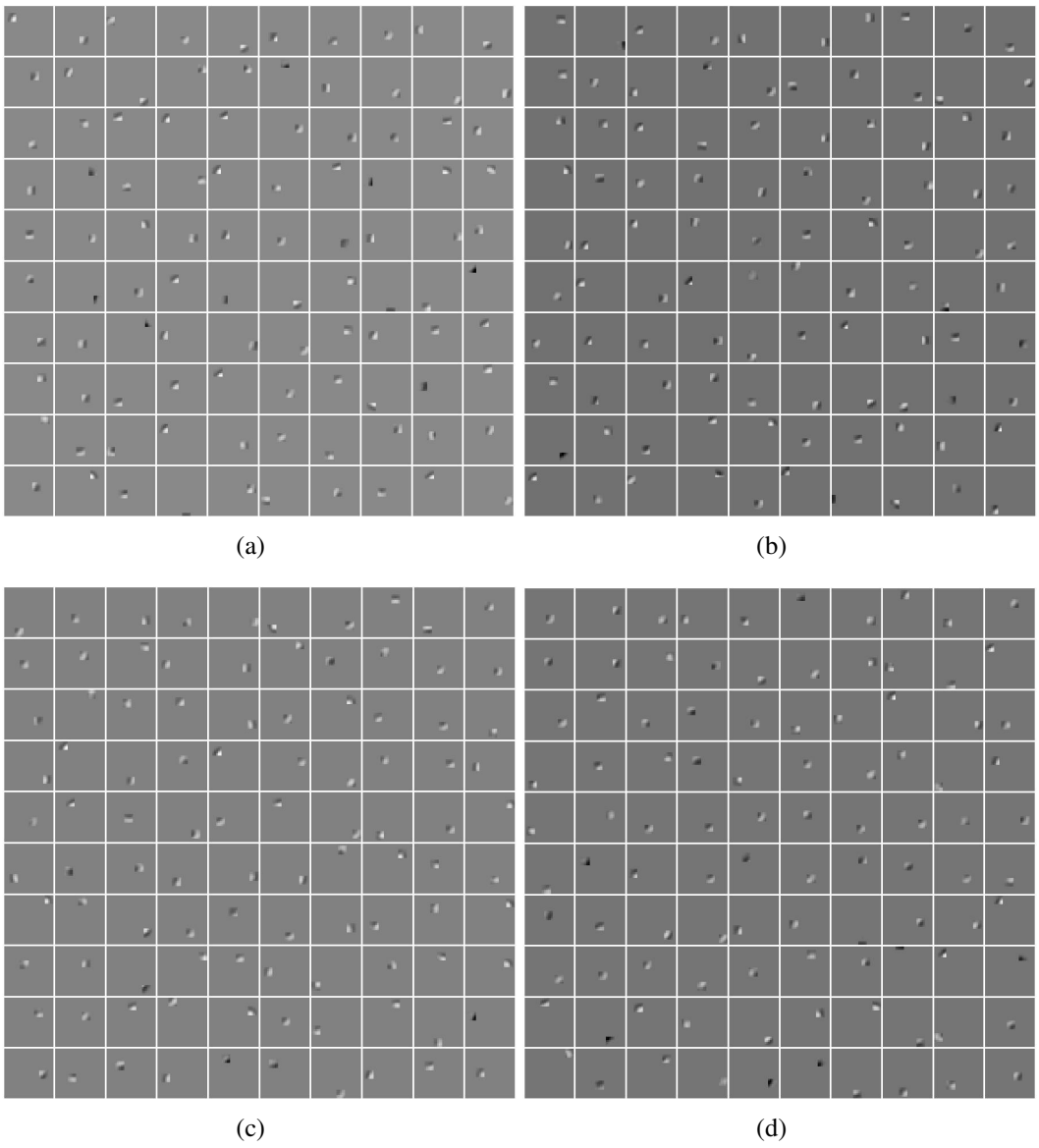


Figure 4.2: Gabor-like receptive fields learned from shifted MNIST digits by the first layer of the Hebbian neural network using (a) Foldiak's method; (b) King's method; (c) Static inhibition; (d) Dynamic inhibition.

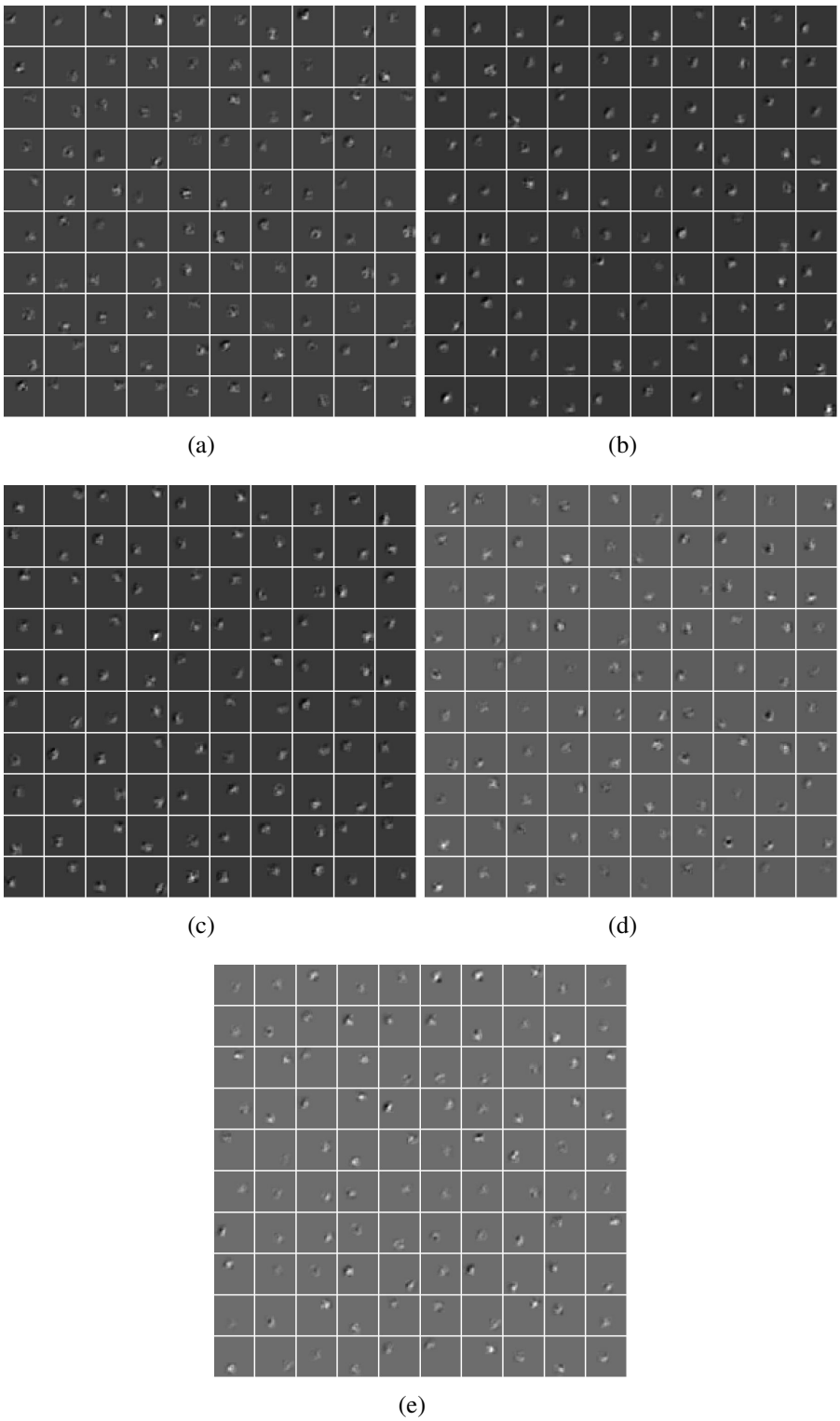


Figure 4.3: Receptive fields learned from shifted MNIST digits by the second layer of the Hebbian neural network using: (a) Földiak's method; (b) King's method; (c) Static inhibition; (d) Dynamic inhibition; (e) Dynamic inhibition with dis-inhibition.

Classification performance

In this section the invariance to shift and rotation and robustness to noise are examined. The tested methods are as follows: 1) Dynamic, 2) Dynamic together with dis-inhibition (DI), 3) Static, 4)Static together with dis-inhibition, 5) a method introduced in King et al. [2013], here called "King" (section 4.1.3) and 6) the famous method introduced in Foldiak [1990], here called "Földiak" (section 4.1.3).

Beside these qualitatively comparable methods, four others are also tested whose low performance helps us to understand the well-performing methods. These four different combinations and their performances are:

1) dis-inhibition on both layers; Performance: First layer: 93.04 % ; Second layer layer: 94.96 % .

2) Földiak method with dis-inhibition on the second layer: First layer: 96.65 % ; Second layer: 95.31 % .

3) King's method with dis-inhibition on the second layer: First layer: 97.13 % ; Second layer: 93.39 % .

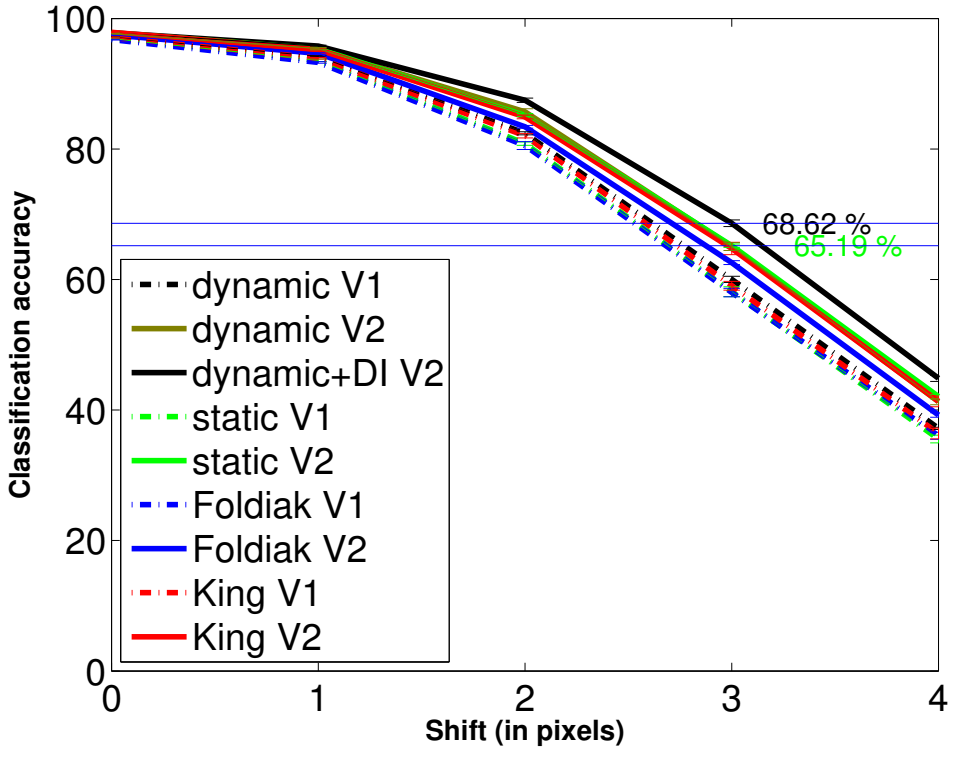
4) Static method on both layers with dis-inhibition on the second layer: First layer: 96.93 % ; Second layer: 90.69 %.

From these unsuccessful experiments, one could conclude that the dis-inhibitory mechanism is not proper beside Static, Földiak and King methods. In the next sections classification results under gradual increase of shift and rotation and also under noise will be explained. The goal was to observe the robustness of different methods against shift, rotation, and noise. In short, as one could see on the following results, using a dynamic learning rule on the first and second layer together with a dis-inhibitory mechanism on the second layer brought the best performance in the sense of robustness.

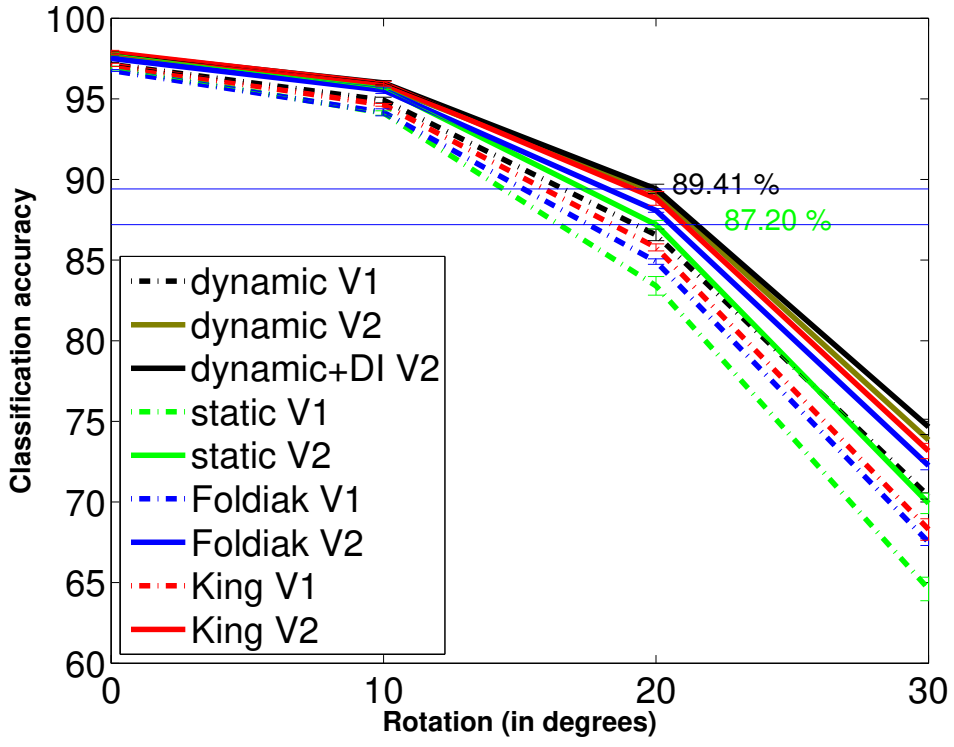
Shift and rotation invariance

In order to measure the robustness of the network against shift and rotation, all methods were tested on the shifted and rotated MNIST. One could observe on figure 4.4 and figure 4.5 that the dis-inhibition mechanism considerably increases the robustness in

classification accuracy. The classification performance with Dynamic, Dynamic+DI, King, Földiak methods are also shown in tables 4.1 and 4.2.

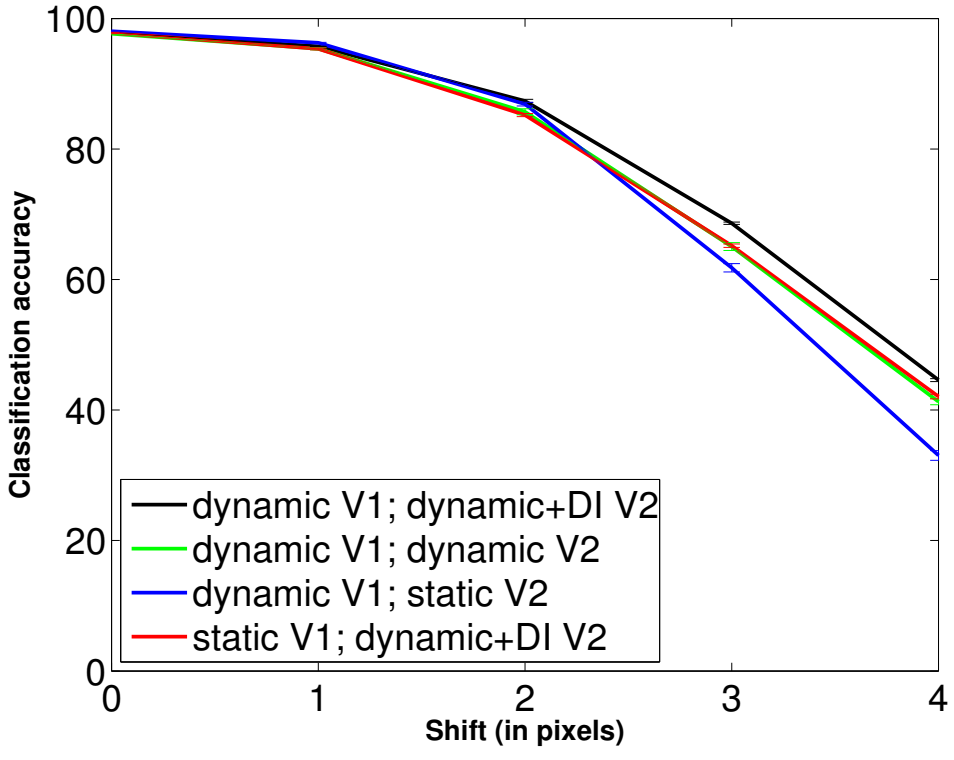


(a)

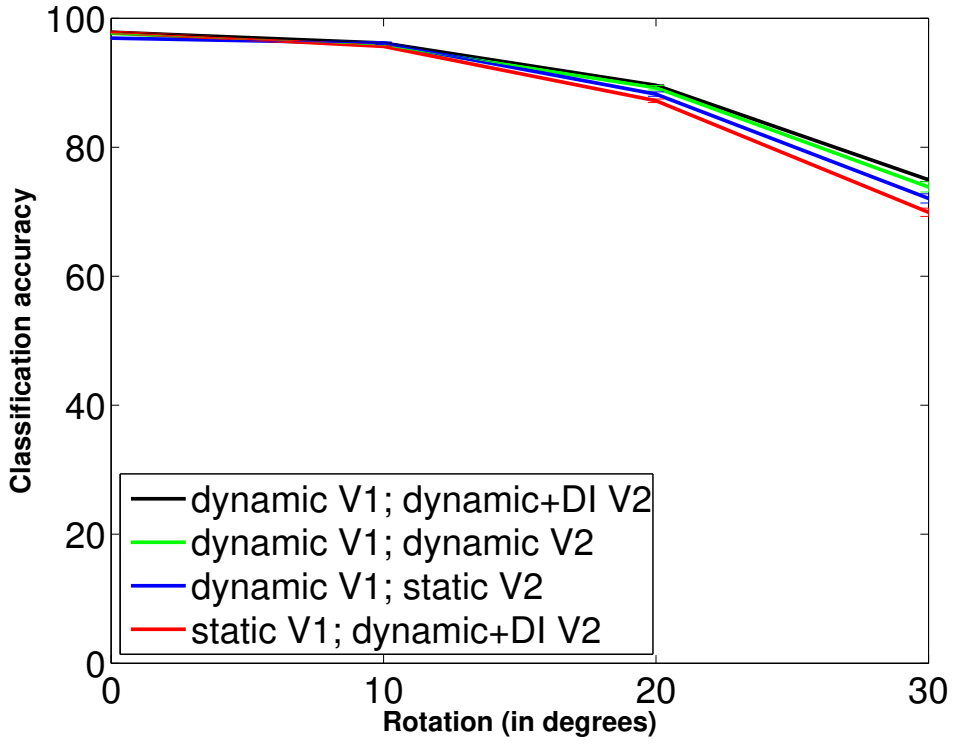


(b)

Figure 4.4: Invariance of Static, Dynamic+dis-inhibition, Földiak and King methods on the first (dotted lines) and second (solid lines) layers. Figure (a) shows the gradual drop of classification performance under increasing shift on the input and figure (b) illustrates the classification accuracy under gradual increase of the rotation of the input. V1 stands for the first and V2 for the scond layer of the network.



(a)



(b)

Figure 4.5: Invariance of alternative combinations of static and dynamic inhibition on the second layer of the network. Figure (a) shows the gradual drop of classification performance under increasing shift on the input and figure (b) illustrates the classification accuracy under gradual increase of the rotation of the input.

Table 4.1: Results of classification on the first (V1) and second (V2) layer of the neural networks under MNIST with the gradual increase of shift from 0 (no shift) to 4 pixels. The dynamic method together with dis-inhibition (Dynmaic+DI) and the King's method showed highest invariance to shift

Method	V1/V2	0 pixels	1 pixel	2 pixels	3 pixels	4 pixels
Dynamic+DI	V1	$97.14 \pm 0.12\%$	$94.19 \pm 0.08\%$	$82.41 \pm 0.24\%$	$60.02 \pm 0.47\%$	$37.18 \pm 0.17\%$
	V2	$97.81 \pm 0.09\%$	$95.78 \pm 0.14\%$	$87.47 \pm 0.33\%$	$68.62 \pm 0.51\%$	$44.85 \pm 0.48\%$
King	V1	$97.10 \pm 0.10\%$	$93.90 \pm 0.12\%$	$82.03 \pm 0.33\%$	$59.05 \pm 0.68\%$	$36.41 \pm 0.82\%$
	V2	$97.87 \pm 0.13\%$	$95.24 \pm 0.20\%$	$84.91 \pm 0.28\%$	$64.75 \pm 0.95\%$	$41.33 \pm 0.83\%$
Dynamic	V1	$97.10 \pm 0.12\%$	$94.21 \pm 0.06\%$	$82.44 \pm 0.37\%$	$59.78 \pm 0.57\%$	$36.71 \pm 0.28\%$
	V2	$97.67 \pm 0.03\%$	$95.38 \pm 0.17\%$	$85.70 \pm 0.46\%$	$65.04 \pm 0.60\%$	$41.278 \pm 0.47\%$
Static	V1	$96.90 \pm 0.12\%$	$93.60 \pm 0.19\%$	$80.81 \pm 0.26\%$	$58.13 \pm 0.75\%$	$35.56 \pm 0.58\%$
	V2	$97.80 \pm 0.08\%$	$95.35 \pm 0.14\%$	$85.21 \pm 0.24\%$	$65.19 \pm 0.26\%$	$42.05 \pm 0.36\%$
Földiak	V1	$96.76 \pm 0.06\%$	$93.27 \pm 0.11\%$	$80.52 \pm 0.60\%$	$58.00 \pm 0.66\%$	$36.04 \pm 0.53\%$
	V2	$97.51 \pm 0.09\%$	$94.63 \pm 0.16\%$	$83.40 \pm 0.20\%$	$62.58 \pm 0.30\%$	$39.23 \pm 0.35\%$

Table 4.2: Results of classification on the first (V1) and second (V2) layer of the neural networks under MNIST with gradual increase of rotation from 0 (no rotation) to 30 degrees. The dynamic inhibition method together with dis-inhibition (Dynamic+DI) and the King's method showed the highest invariance to rotation

Method	V1/V2	0 degree	10 degrees	20 degree	30 degrees s
Dynamic+DI	V1	$97.14 \pm 0.1219\%$	$94.93 \pm 0.18\%$	$86.56 \pm 0.36\%$	$70.37 \pm 0.20\%$
	V2	$97.81 \pm 0.09\%$	$95.10 \pm 0.14\%$	$89.41 \pm 0.30\%$	$74.66 \pm 0.48\%$
King	V1	$97.10 \pm 0.11\%$	$94.63 \pm 0.085\%$	$85.79 \pm 0.22\%$	$68.29 \pm 0.66\%$
	V2	$97.87 \pm 0.13\%$	$95.94 \pm 0.15\%$	$88.83 \pm 0.42\%$	$73.16 \pm 0.47\%$
Dynamic	V1	$97.10 \pm 0.12\%$	$94.64 \pm 0.12\%$	$86.80 \pm 0.40\%$	$69.86 \pm 0.53\%$
	V2	$97.67 \pm 0.034\%$	$95.79 \pm 0.15\%$	$89.22 \pm 0.34\%$	$73.85 \pm 0.79\%$
Static	V1	$96.90 \pm 0.12\%$	$94.12 \pm 0.14\%$	$83.40 \pm 0.58\%$	$64.61 \pm 0.73\%$
	V2	$97.80 \pm 0.08\%$	$95.64 \pm 0.06\%$	$87.20 \pm 0.27\%$	$69.91 \pm 0.63\%$
Földiak	V1	$96.76 \pm 0.06\%$	$94.17 \pm 0.22\%$	$84.91 \pm 0.17\%$	$67.51 \pm 0.21\%$
	V2	$97.51 \pm 0.09\%$	$95.54 \pm 0.20\%$	$88.08 \pm 0.11\%$	$72.25 \pm 0.25\%$

Figure 4.4 (a) shows the classification performance under gradual increase of shift in the input. On the first layer (dotted lines) the dynamic methods (black) showed slightly better performance than the others, while the static and Földiak methods were the weakest ones. The King method took the second best place. On the second layer (solid lines) the dynamic together with dis-inhibition showed a considerably better robustness. On the second layer, the static and King methods had the middle-level performance, and the Földiak method was the weakest. Figure 4.4 (b) shows the classification performance under gradual increase in rotation in the input. Here the static method showed the lowest performance on both layers while the dynamic plus dis-inhibition was the superior method. The King method took the second best place and the Foliak method was the second worst one.

Figure 4.5 illustrates the performance of alternative combinations of static and dynamic inhibition. Just the second layer performance is shown. Figure (a) shows the classification performance under gradual increase of shift in the input. The best performance was achieved when both layers employed the dynamic mechanism and the second layer benefited from dis-inhibitions (black). The performance was less when the first layer instead employed a static inhibition (red). A lower invariance was achieved in the case both layers employed dynamic inhibition but the second layer did not use a dis-inhibitory mechanism. The worst performance was achieved if the first layer used a dynamic but the second layer a static learning rule for the inhibitory weights. Figure (b) illustrated the classification performance under the gradual increase in the angle of rotation of the input. Similar to the experience with shift, the worst performance was achieved in the case the first layer was dynamic but the second layer static. The other two combinations took the same place after the dynamic+dis-inhibition and the worst case.

Robustness to noise

To study the effect of dynamic inhibition and dis-inhibition in overcoming distortions in the input, the network was also tested on a noisy version of MNIST: n-MNIST. n-MNIST contains three sets of modified MNIST digits with Additive White Gaussian Noise (AWGN), Motion Blur (MB) and with reduced contrast and AWGN (RC and AWGN) (`http://csc.lsu.edu/~saikat/n-mnist/`). Examples of n-

MNIST images are shown in figure 4.6. Table 4.3 shows the mean classification performance of 5 experiments when n-MNIST was fed to the networks.

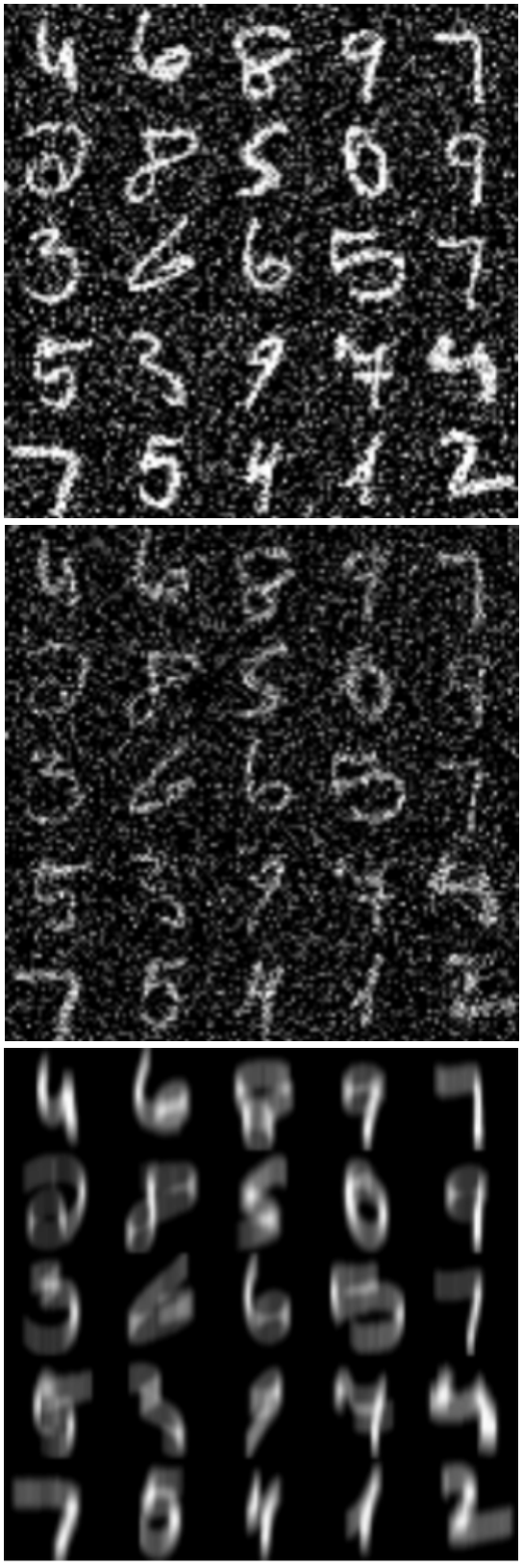


Figure 4.6: Illustrations of raw (before whitening) MNIST images (a)MNIST with Additive White Gaussian Noise (AWGN) (b)MNIST with reduced contrast and AWGN (c)MNIST with Motion Blur

Table 4.3: Results of classification on the first (V1) and second (V2) layer of the neural networks under noisy MNIST (n-MNIST).

Method	*Layer*	Dynamic+DI.	Static	Földiak	King
AWGN	V1	$94.582 \pm 0.14\%$	$94.44 \pm 0.14\%$	$94.10 \pm 0.05\%$	$94.67 \pm 0.15\%$
	V2	$95.016 \pm 0.07\%$	$95.26 \pm 0.12\%$	$94.67 \pm 0.09\%$	$95.23 \pm 0.17\%$
RC-AWGN	V1	$90.57 \pm 0.20\%$	$90.46 \pm 0.15\%$	$89.67 \pm 0.20\%$	$90.43 \pm 0.28\%$
	V2	$90.66 \pm 0.19\%$	$90.96 \pm 0.22\%$	$89.90 \pm 0.21\%$	$90.66 \pm 0.31\%$
MB	V1	$97.35 \pm 0.08\%$	$97.10 \pm 0.05\%$	$97.16 \pm 0.094\%$	$97.24 \pm 0.06\%$
	V2	$98.12 \pm 0.10\%$	$98.08 \pm 0.08\%$	$98.09 \pm 0.10\%$	$98.00 \pm 0.17\%$

None of the methods showed a considerable advantage to the others, although the Földiak method performed weaker than the others. The surprising result was that under motion bur (MB) all methods performed at-least as good as the case without noise.

Histogram of lateral weights

As the main goal of this study is to find a way to get rid of the inhibitory weights in the cases it causes loss of information, it is interesting to see which range of lateral weights are more harmful. Thus, the histograms of the lateral weights were visualized (figure 4.7). One could see that the higher the number of the middle range lateral weights of the first layer, the lower is the performance of the network. That is, the weights which cause the loss of information could be the middle range weights. The other fact is that in the successful methods, at the end of the learning process most of the lateral weights are set to zero and there are few ones which have indeed strong strength. This is due to the fact that the lateral weights exist to force the neurons to learn vectors with directions as different as possible: as soon as the lateral weights have done their job, they should get disappeared else they would send inhibitory signals to neurons which do not have strong correlation with other active neurons. In all methods, the lateral weights of the second layer have lower values than on the first layer. As the histogram shows, the DI removes a part of the middle range weights of the second layer when used beside the

Dynamic method. This is again in line with the higher robustness of the Dynamic+Di in comparison to the pure Dynamic method.

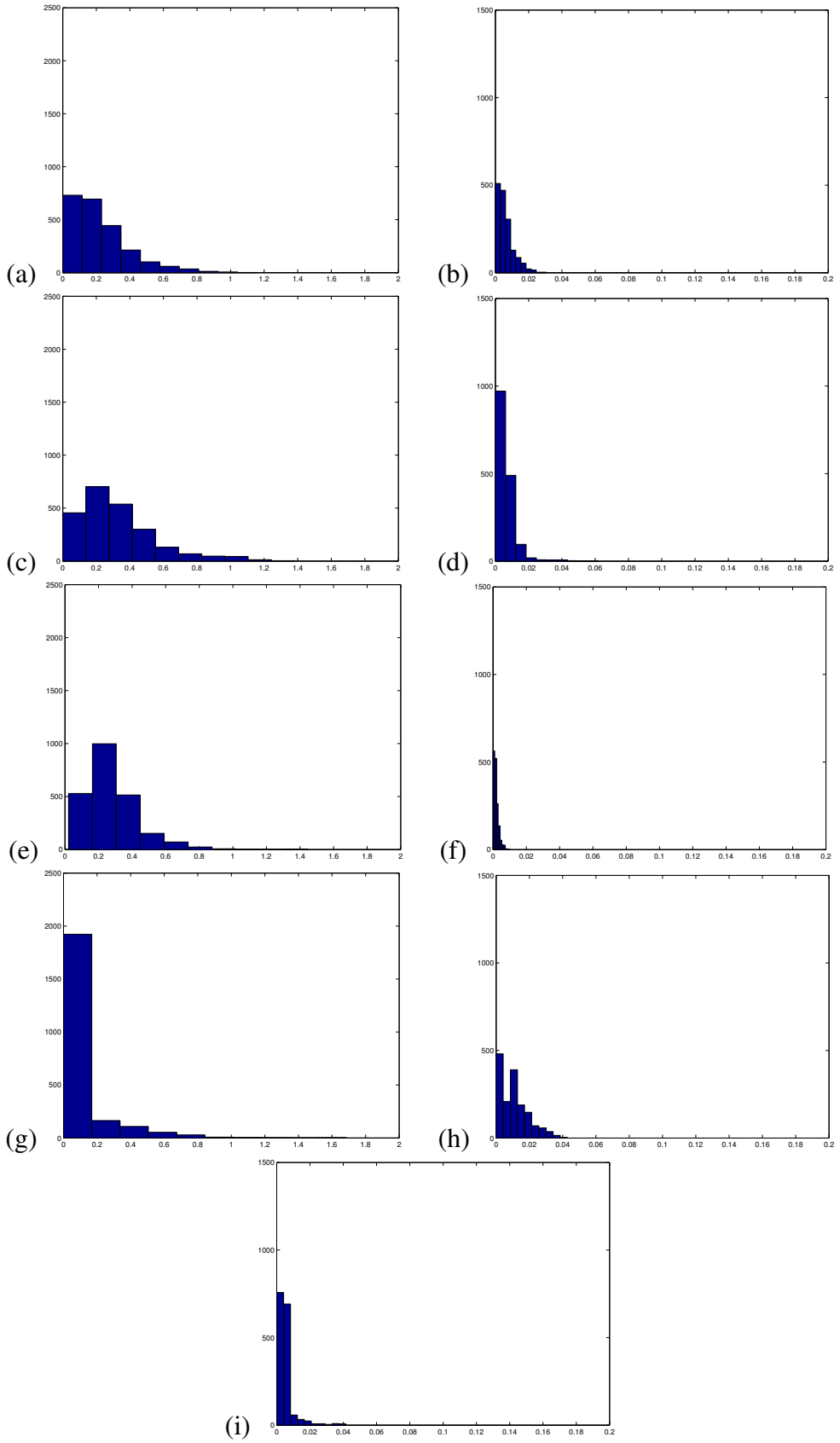


Figure 4.7: Histogram of lateral weights in ... (continued on the following page)

Figure 4.7: (continued from the previous page) (a)First layer of King's method (b) Second layer of King's method (c)First layer of Földiak inhibition method (d) Second layer of Foldiak inhibition method (e)First layer of the static inhibition method (f)Second layer of the static inhibition method (g)First layer of dynamic inhibition method (h)Second layer of dynamic inhibition method (i)Second layer of dynamic inhibition method together with dis-inhibition (DI).

Entropy and negative activities

In order to investigate how valid is the hypothesis that dynamic inhibition prevents loss of information, some statistical analysis were applied to the output of the networks having different inhibitory mechanisms including the dynamic inhibition. First of all, the percentage of negative membrane potentials each method causes was measured. This was done by dividing the absolute sum of negative activities by the total sum of the activities of all neurons on a specific layer under the 3-pixel shifted MNIST test set. As it was shown in section 4.3, the difference in performance of different inhibitory mechanisms is mostly visible under three pixels shift. That was why the 3-pixel shifted test set was chosen for further statistical analysis.

Beside the percentage of the negative membrane potential, the entropy of the output activities was also measured on the same set of inputs. By definition, entropy is a measure for information (Shanon information). In other words, higher entropy should mean greater amount of information contained in the signal. To investigate if also the population activities of the networks studied in this work follow this hypothesis, the classification accuracy of the activities was also included in the measurements.

As one could observe on table 4.4, the amount of entropy and the percentage of the negative activity correspond with the accuracy of classification. In other words, **higher** entropy, and **lower** negative activity occur at the same time with **higher** classification rate. One could conclude that the dynamic inhibitory mechanism lets more amount of information (entropy) to appear on the output by preventing the activities from being rectified. The meaningfulness of information is measured with the classification accuracy.

Table 4.4: Statistical aspects of the algorithms on original MNIST test set with three pixels random shift.

	Layer	Dynamic+DI	Dynamic	Static	King	Földiak
Negative MP.	V1	$46.64 \pm 0.50\%$	$46.62 \pm 0.62\%$	$70.28 \pm 0.89\%$	$74.14 \pm 0.27\%$	$79.13 \pm 0.21\%$
	V2	$2.83 \pm 0.38\%$	$6.09 \pm 0.45\%$	$20.20 \pm 0.45\%$	$55.41 \pm 3.52\%$	$35.83 \pm 1.04\%$
Entropy	V1	$0.97 \pm 0.01\%$	$0.97 \pm 0.003\%$	$0.69 \pm 0.01\%$	$0.66 \pm 0.004\%$	$0.61 \pm 0.001\%$
	V2	$1.94 \pm 0.02\%$	$1.41 \pm 0.03\%$	$0.56 \pm 0.06\%$	$0.09 \pm 0.02\%$	$0.27 \pm 0.04\%$
Class. acc.	V1	$60.02 \pm 0.47\%$	$59.78 \pm 0.57\%$	$58.13 \pm 0.75\%$	$59.05 \pm 0.68\%$	$58.00 \pm 0.66\%$
	V2	$68.62 \pm 0.51\%$	$65.04 \pm 0.60\%$	$65.19 \pm 0.26\%$	$64.75 \pm 0.95\%$	$62.58 \pm 0.30\%$

Negative activities (NA) are the membrane potential (MP) values which are below zero and which are set to zero by the algorithm. Although this rectification is necessary for biological plausibility, it removes some information about the input which could be passed to the next layers and from which the neurons could learn. As it was expected, Dynamic inhibition produced the least amount of NA, with around 1.6 times less NA in comparison with other methods on the first layer. On the second layer, using just Dynamic, NA is up to 9.09 times less than the other methods and if Dynamic Inhibition used together with Dis-Inhibition (Dynamic+DI), NA is up to 19.57 times less than that of the other methods.

As shown on the table 4.4, the entropy of the code produced by Dynamic inhibition is up to 1.6 times on the first layer and 15.66 times on the second layer more than the entropy of the other methods. Dynamic+DI produces outputs with up to 21.5 times higher entropy than the others. This is in line with the variation of NA, which shows that the rectified negative activities actually contain information. As shown on the table, lower negative activities and higher entropy coincide with higher classification accuracy (Class. acc.).

4.4 Discussion

As it was said before, because the competition in the mentioned systems is achieved using inhibition, a strong inhibition could theoretically guarantee strong difference among the features, but if the inhibition is too strong, it could kill out the activities. So, one should find an optimal amount of inhibition which could separate the features as much as possible but prevent suppressing the activities in the point where no further separation is possible.

In this section the hypothesis based on which the dynamic inhibition and also the dis-inhibition were invented and also the results of simulations will be discussed. As the main idea behind the dynamic inhibition was preventing membrane potential from going negative and being rectified, the percentage of negative membrane potential activities was an important measure for evaluating the hypothesis. The idea of preventing negative membrane potential itself comes from the need for preventing loss of informa-

tion. The theory was that if a big portion of the membrane potential is rectified by the activation function, some information contained in those rectified activities will be lost as in fact, membrane potential is a function of the input and contains information about it. In order to make sure that the rectified part of the membrane potential contains information, the entropy was measure on the activities of the neurons. It was observed that when the percentage of the negative/rectified membrane potential is lower, the entropy of the output is higher meaning more information about the output is contained in the population code. In order to investigate if this information is meaningful, the output was classified using a linear classifier and the accuracy was measured. The higher classification rate coincided with higher entropy, approving that the information produced by the networks is describing the input. As shown in Table 4.4, the lower amount of negative membrane potential coincides with higher entropy and higher classification rate when the input is shifted for three pixels. The observed higher classification rate under dynamic inhibition is an evidence of good coding and low amount of negative membrane potential together with the high entropy are evidences of low level of information loss. So, one could conclude that among the studied methods, dynamic inhibition together with dis-inhibition builds up the superior system.

Further, the dynamic inhibition and the dis-inhibitory mechanism are discussed.

4.4.1 Dynamic inhibition and Dis-Inhibition

There exist several similar models to the one suggested in section 4 with an inhibitory mechanism based on Hebbian learning; starting from Foldiak [1990] with rather simpler mechanism to the more recent and sophisticated methods like King et al. [2013] which use inter-cellular inhibitory cells for inducing competition.

It is obvious that if we feed several neurons with a similar input without any competition/inhibition mechanism, the learned components will be mostly similar, strongly overlapping and will contain redundant information. Thus, competition is employed not to let neurons to have the similar activities, that is, not to let them learn with the same strength from a single input. As a result, neurons have different activation levels under the same input and thus, the actual input has different influences on the synaptic weights of different neurons. This happens because although their weights have the same pre-

synaptic values, the Hebbian learning rule updates the weights based also on different post-synaptic values. As a result, the winning neuron learns faster than the losers from a specific input.

With the applied modification to the traditional anti-Hebbian learning mentioned in the section 4.2, the weight reduction mechanism is not trying to normalize the weight matrix like in Oja [1982], but attempts to balance the inhibition and excitation to prevent loss of information. This method is also different from Teichmann et al. [2012] and Kermani Kolankeh et al. [2015] which are the base of the current work but use a static reduction coefficient for the inhibitory weights. In the current model, the dynamic weight reduction factor for the inhibitory weights regulates the strength of the inhibition a neuron receives. This is done by subtracting $\omega = (net^{post} - mp^{post})^2/\eta$ from the inhibitory weights. It forces the system to have the equilibrium at the point when the membrane potential and the net activity of the neuron are equal. This will happen when there is no inhibition on the neuron which could push the membrane potential to the values less than the net activity. On the one hand this will give the inhibition a limited time to do its job as the neuron tends to get rid of the inhibition. If the inhibition is not effective enough to decorrelate neurons, the constant increase of the lateral weights should be prevented to avoid killing activities. On the other hand, if the inhibition is already too strong and is pushing the membrane potential to negative values, ω will increase faster as a result of negative mp^{post} and prevents the activities from being negative and rectified by the activation function. This two-sided effect of the dynamic inhibition makes it possible for the system to learn overlapping features when separating them is not possible.

The classification performance on the output of the networks in when the original (not shifted and not rotated) digits were feed to the networks was slightly weaker in the first layer in Foldiak and Static but in the Dynamic, Dynamic+DI and King the performance was almost the same. In the second layer also all methods showed almost the same classification accuracy. The difference in performance was visible when some rotation or shift was applied to the input. As one could see on the tables 4.1 , 4.2 or Figure 4.4, the maximum difference in the performance is under 3 pixels shift and 30 degrees rotation visible. Although the dynamic method causes higher classification performance on both layers, employing DI together with it, increased the performance by 4% under three pixels shift and 1% under 30 degrees rotation. Returning to the goal of introducing

the Dynamic method, classification accuracy was not the main goal of this research. It could be considered as a side effect of better coding as a result of higher quality of feature learning or as the result of lower information loss. In any case, from the fact that higher entropy achieved using Dynamic method has not caused lower classification accuracy, it could be concluded that the entropy is not the result of noisy information. One could see (Table 4.4) that the Dynamic inhibition causes around 30% less negative membrane potential than the other methods on the first layer and up to 49% less on the second layer. In addition, when DI accompanied Dynamic inhibition, the native MP on the second layer dropped from 6% to around 3%. These lower amounts of negative MP achieved using Dynamic and Dynamic+DI coincided with higher entropy. This tells about the fact that the negative membrane potential which could potentially be rectified out, is not uniform and contains a considerable variety of values causing higher entropy. As it was mentioned, the fact that this part of the signal at least did not confuse the classifier is a sign that the rectified part of the signal could contain meaningful information.

A dis-inhibitory mechanism, inspired from the VIP dis-inhibitory neurons in the brain, was designed with the purpose of aggregation of information and achieving invariance on the second layer of the network. Although the effect of dis-inhibition in the small neighborhoods produced by VIP cells in the brain is not clearly understood, in this work it is proposed that what dis-inhibition does is, in fact, announcing one of the cells in a neighborhood as the winner to induce a winner-take-all mechanism. This WTA mechanism gives one cell the chance of learning an aggregation of the features which its neighbors together could learn from the current input. This means that the winner neuron will respond to all of these features and thus will be invariant to the details which would have made the features different. In the proposed model for Dis-Inhibition, the most active neuron in a neighborhood is declared as the winner by removing the inhibitory weights from the other cells to it. As the result, the winner will be capable of inhibiting the other neurons but not vice versa.

The advantage of using dis-inhibition is observable in all criteria. That is, in the percentage of the negative activity, in the entropy and the classification accuracy. As it was shown is the section 4.3 the dis-inhibitory mechanism causes removal of many inhibitory weights and obviously with lower number of inhibitory weights remained after

the learning phase, less inhibition will be applied to the cells. Naturally, this causes less amount of negative activities. It was observed that under dis-inhibition, also the entropy was maximal.

The effect of dynamic inhibition in the first layer on the performance of the second layer could also be observed on figure 4.5 (red line). As mentioned in 4.3, if both dynamic inhibition and dis-inhibition are used on the second layer without employing dynamic inhibition on the first layer, although we will achieve good results with the non-shifted and not rotated digits, the robustness of the network to shift and rotation will be decreased on the second layer (in comparison to the case of using dynamic inhibition on the first layer). This means that the information saved on the first layer by the means of dynamic inhibition is used on the second layer for a better feature extraction. In addition to this fact, the difference between the performance of the static and dynamic methods on the first layer is not as visible as on the second layer. This is because the overlapping features learned on the first layer contain information which perhaps not helpful for the classifier on the first layer, but could be decoded by the second layer and passed to the output.

Figure 4.5 also shows that the dynamic inhibition on the first layer alone does not cause a good performance on the second layer if no dynamic and dis-inhibition mechanisms are used on the second layer (blue line). This could mean that the overlapping features of the first layer need to be separated on the second layer by the dynamic inhibition (green line) and aggregated (black line) by the dis-inhibitory mechanism.

Strong aggregation in the lower layers could disable them from learning overlapping features by different neurons. This was tested by applying dis-inhibition to the first layer of the network. It not only didn't cause any improvement but also worsened the performance. On the other hand, the same mechanism was useful on the second layer. This could also suggest that we need the first layer to learn overlapping features by letting the neurons to have partial co-activation. Although, on the second layer we could already aggregate the information from the overlapping features and achieve invariance.

Aggregation of information by dis-inhibition could be harmful if it does not cause invariance, but loss of information. The dynamic inhibition without dis-inhibition causes a more dense code than the other methods and this dense code brings confusion to the clas-

sifier and respectively less performance. Although if used together with dis-inhibition, the dynamic inhibition shows the best performance. This means that dis-inhibition extracts valuable information out of the dense code of dynamic inhibition method. On the other hand, when dis-inhibition applied to the first layer together with the dynamic inhibition, it leaves less valuable information for the second layer meaning that on the first layer we need yet to learn overlapping features carrying valuable information to be decoded on the second layer (4.3).

From one point of view, the dis-inhibition mechanism removes unnecessary inhibitory weights which could cause loss of information. According to the histogram of lateral weights of the second layer depicted in figure 4.7, the lower performance of the weaker algorithms coincides with the existence of middle range lateral weights. In better algorithms, the lateral weights are mostly zero and with a limited number of strong weights. When comparing images (h) and (j) in figure 4.7, one can see that employing dis-inhibition can remove midrange weights which are the result of competition among similar (overlapping) feature.

In order to investigate how much role the dis-inhibition plays in robustness against noise, n-MNIST, a noisy version of MNIST was tested on the mentioned algorithms. As one could observe in table 4.3, none of the methods were superior, although the Földiak method showed slightly worse performance than the others. It was not expected that a mechanism comparable to pooling would improve noise invariance in addition to invariance against shift and rotation. The fact that motion blur increased the performance of the network on the first and second layers is a clue that the competition is playing role in discrimination. As an example, looking at blurred digit 7, one could say that it is actually several 7s together could activate both neurons which detect components of 7 but also neurons which are sensitive to the components of digit 1 but obviously these neurons have been losers of the competition, that is why the performance is even better than the case there was no noise on the input (4.3).

The invariance dis-inhibition brings to the network could be compared with the invariance pooling brings to multiplayer convolution networks. The similarity is in the fact that both methods privilege one neuron in a neighborhood to the other neurons and let it learn the aggregation of the information that the neurons in the neighborhood would have learned. Although the main idea of pooling comes from the function of com-

plex cells which cause invariance in the nervous system, the invariance achieved by dis-inhibition could not be an explanation for the function of complex cells.

Chapter 5

Conclusion and future perspectives

Each neuron is a feature detector transferring the data into a new space. The new space is at risk of confusion if these feature detectors overlap. Here overlapping means being similar; that is, overlapping in the feature space. Overlap avoidance or separation of features describing the space is done with the help of suppression. That is, to prevent a group of neurons from learning the same pattern, they are suppressed by each other, preventing each other from learning at the same time. It was shown that this prevention not only plays a role in a better classification of objects, it is also helpful for overcoming confusion made by loss of information in the input. This is because a good separation of objects in the new space makes it difficult for an object to be classified in a wrong group even if experiences fluctuations. However, the avoidance of overlap among features which helps the objects to stay separated in the new space should not be achieved at the cost of losing information. Loss of information caused by heavy inhibition could be explained in following points:

a) The unnecessary appearance of inhibition could suppress meaningful low activities based on which neurons could learn from the input. This causes a too sparse population code not carrying enough information for the upper layers.

b) Too much inhibition from the winner neuron could keep the loser inactive even if the winner is not highly active. Babenko and Lempitsky [2015] explains this case as follows: "Inefficient inhibition can cause suppression of true positive together with false positive". In other words, any considerable activity of the winner neuron could cut out the rest of the neurons in the neighborhood and decrease the capacity of the network.

c) Loss of information happens when the inhibitory signal a neuron receives is bigger than the excitation it receives and the membrane potential is pushed to negative values. Because a neuron can not have a negative activity, the rectifier in the activation function clips the activity to zero and although the membrane potential of the neuron is changing as a function of the input, this change would not be observable on the output.

So there should be a balance between inhibition and excitation which keeps a neuron alive but does not let it learn redundant information which has already been learned by the other units. In other words, unnecessary inhibition should be avoided. Based of the point c) mentioned above, it is tried to solve this problem by decreasing the inhibitory weights when the membrane potential of a neuron is lower than its net value (weighted sum of its inputs). The idea behind this is that what makes the membrane potential lower than the net value is the inhibition. If the inhibition is not capable to decorrelate two units in a short time, it should be reduced to let the neurons learn overlapping features instead of constantly increasing the inhibitory weights. This way, loss of information by too heavy inhibition is avoided.

It was observed in this study that a dis-inhibitory mechanism which favors the winner of the competition by removing inhibition to it, could cause invariance to shift and rotation. When using dis-inhibition, the winner learns based on an aggregation of the similar stimuli while the loser is suppressed and is prevented from learning from the current input. This could remind one of the complex cells introduced in Hubel and Wiesel [1982] and its extensions to different pooling mechanisms (Boureau et al. [2010]) which were later also used in Convolutional Neural Networks (CNN) (LeCun et al. [1998]). These mechanisms have the job of aggregation after the separation which is done on the lower layers. In other words, after the lower layer extracts the features which could describe the input, the complex or pooling layer, brings a level of invariance by aggregating the similar features. In the proposed model though, the second layer is not purely an aggregation layer, but also a feature extractor similar to the first layer.

Dis-inhibition could have its biological roots in the VIP cells in the brain which have the job of dis-inhibiting the pyramidal excitatory cells. VIP cells do this by inhibiting the SOM cells which in turn have the responsibility of inhibiting pyramidal cells. One may make the hypothesis that perhaps the dis-inhibited pyramidal cells are actually the winners of the competition. The small areas VIP cells cover also could be an evidence

of the fact that they target the winners to induce some kind of winner-take-all and also as suggested in our model, to bring some invariance to the system. As a future task, this hypothesis could be tested experimentally by biologists.

Both dynamic inhibition and dis-inhibition are based on the same principle: avoiding extra inhibition. The dynamic inhibition does it with reducing the weights when they are causing loss of information or when they are not helpful. Dis-inhibition does it with complete removal of inhibition in specific conditions when inhibition is harmful. In the proposed two-layer model the dis-inhibitory system is used just on the second layer meaning that the proposed VIP-like cells are affecting just the second layer. As a future improvement to the model, one could change the dis-inhibitory mechanism in a way that it could exist on each layer.

Currently, the network has a huge number of cells which could be reduced with the goal of increasing the speed. Much effort on this work was on the dynamic avoidance of extra inhibition. One could look for more appropriate architectures with less number of neurons. In this improvement, one could seek the optimal size for receptive fields of both excitatory and inhibitory cells. The number of maps on each layer also is of great importance.

In the existing model more than two layers are not helpful as the result of the fact that the network does not follow any global goal and the learning is unsupervised. A feedback signal based on the labels of the input could guide the network toward the goal of better classification and make it possible to learn features which could contain more useful information for the next layers.

As a future work, one could additionally measure the degree of overlap among features to see how much overlap is necessary and how much is harmful. The dynamic inhibition and dis-inhibition both support overlap. Th hypothesis is that on the one hand, dynamic inhibition lets neighboring neurons to have overlapping features when separation causes loss of information and on the other hand, the dis-inhibition gives the winner the ability to learn overlapping features. This hypothesis is tested indirectly by measuring the classification rate. In addition, one could **directly** investigate how much the features overlap under different scenarios. This could help to make a better dynamic mechanism for preventing extra inhibition.

Bibliography

Ossama Abdel-hamid, Li Deng, and Dong Yu. Exploring Convolutional Neural Network Structures and Optimization Techniques for Speech Recognition. *14th Annual Conference of the International Speech Communication Association, (INTERSPEECH 2013)*, (August):3366–3370, 2013.

Daniel J Amit and N Brunel. Model of global spontaneous activity and local structured activity during delay periods in the cerebral cortex. *Cerebral cortex (New York, N.Y. : 1991)*, 7(3):237–252, 1997. ISSN 1047-3211. doi: 10.1093/cercor/7.3.237.

Pradeep Arkachar and Meghanad D. Wagh. Criticality of lateral inhibition for edge enhancement in neural systems. *Neurocomputing*, 70(4-6):991–999, 2007. ISSN 09252312. doi: 10.1016/j.neucom.2006.03.017.

Artem Babenko and Victor Lempitsky. Aggregating Local Deep Features for Image Retrieval. *Proceedings of the IEEE International Conference on Computer Vision*, pages 1269–1277, 2015. ISSN 978-1-4673-8391-2. doi: 10.1109/ICCV.2015.150.

Anthony J. Bell and Terrence J. Sejnowski. The "independent components" of natural scenes are edge filters. *Vision research*, 37(23):3327–38, dec 1997. ISSN 0042-6989.

Yoshua Bengio. Learning Deep Architectures for AI. *Foundations and Trends® in Machine Learning*, 2(1):1–127, 2009. ISSN 1935-8237. doi: 10.1561/2200000006.

Elie Bienenstock, Leon Cooper, and P. Munro. Theory for the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex. *The Journal of neuroscience*, 2(1):32–48, 1982. ISSN 15537366.

Y. Lan Boureau, Jean Ponce, and Yann LeCun. A Theoretical Analysis of Feature Pooling in Visual Recognition. *Icml*, pages 111–118, 2010. doi: citeulike-article-id: 8496352.

Y. Lan Boureau, Nicolas Le Roux, Francis Bach, Jean Ponce, and Yann Lecun. Ask the locals: Multi-way local pooling for image recognition. *Proceedings of the IEEE International Conference on Computer Vision*, pages 2651–2658, 2011. ISSN 1550-5499. doi: 10.1109/ICCV.2011.6126555.

A. N. Burkitt. Balanced neurons: Analysis of leaky integrate-and-fire neurons with reversal potentials. *Biological Cybernetics*, 85(4):247–255, 2001. ISSN 03401200. doi: 10.1007/s004220100262.

Frances S. Chance, L. F. Abbott, and Alex D. Reyes. Gain modulation from background synaptic input. *Neuron*, 35(4):773–782, 2002. ISSN 08966273. doi: 10.1016/S0896-6273(02)00820-6.

Peter Dayan and Lf Abbott. Theoretical Neuroscience. *Computational and Mathematical Modeling of Neural ...*, 60(3):489–95, 2001. ISSN 0262041995. doi: 10.1016/j.neuron.2008.10.019.

Li Deng. Three Classes of Deep Learning Architectures and Their Applications: A Tutorial Survey. *Research.Microsoft.Com*, 2013. ISSN 2048-7703.

James J. DiCarlo and David D. Cox. Untangling invariant object recognition. *Trends in Cognitive Sciences*, 11(8):333–341, 2007. ISSN 13646613. doi: 10.1016/j.tics.2007.06.010.

Alexey Dosovitskiy, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative Unsupervised Feature Learning with Convolutional Neural Networks. *Advances in Nueral Information Processing Systems 27 (Proceedings of NIPS)*, pages 1–13, 2014. ISSN 0162-8828. doi: 10.1109/TPAMI.2015.2496141.

Michael S Falconbridge, Robert L Stamps, and David R Badcock. A simple Hebbian/anti-Hebbian network learns the sparse, independent components of natural images. *Neural computation*, 18(2):415–29, feb 2006. ISSN 0899-7667. doi: 10.1162/089976606775093891.

Ralph Fehrer and Stefan Feuerriegel. Improving Decision Analytics with Deep Learning: The Case of Financial Disclosures. *arXiv:1508.01993 [cs, stat]*, 2015.

Torres Fernandes, Bruno Jose, George D C Cavalcanti, and Tsang Ing Ren. Lateral inhibition pyramidal neural network for image classification. *IEEE Transactions on Cybernetics*, 43(6):2082–2092, 2013. ISSN 21682267. doi: 10.1109/TCYB.2013.2240295.

Peter Foldiak. Forming sparse representations by local anti-Hebbian learning. *Biological Cybernetics*, 64(2):165–170, 1990. ISSN 03401200. doi: 10.1007/BF02331346.

Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, 1980. ISSN 03401200. doi: 10.1007/BF00344251.

Kunihiko Fukushima. Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural Networks*, 1(2):119–130, 1988. ISSN 08936080. doi: 10.1016/0893-6080(88)90014-7.

Kunihiko Fukushima and Sei Miyake. Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position. *Pattern Recognition*, 15 (6):455–469, 1982. ISSN 00313203. doi: 10.1016/0031-3203(82)90024-3.

Ben Graham. FRACTIONAL MAX-POOLING. pages 1–8, 2015.

Karol Gregor, Arthur Szlam, and Yann LeCun. Structured sparse coding via lateral inhibition. *Nips*, pages 1116–1124, 2011. doi: citeulike-article-id:12223524.

S Grossberg. Competitive learning: From interactive activation to adaptive resonance. *Cognitive Science*, 11(1):23–63, 1987. ISSN 03640213. doi: 10.1016/S0364-0213(87)80025-3.

Fred H. Hamker and Jan Wiltschut. Hebbian learning in a model with dynamic rate-coded neurons: an alternative to the generative model approach for learning receptive fields from natural scenes. *Network (Bristol, England)*, 18(3):249–66, 2007. ISSN 0954-898X. doi: 10.1080/09548980701661210.

Donald Hebb. The Organization of Behavior. A Neuropsychological Theory. 1949. *Wiley, New York*, 1949.

John Joseph Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the United States of America*, 79(April):2554–2558, 1982. ISSN 0027-8424. doi: 10.1073/pnas.79.8.2554.

John Joseph Hopfield, D I Feinstein, and R G Palmer. 'Unlearning' has a stabilizing effect in collective memories, 1983. ISSN 0028-0836.

Patrik O. Hoyer. Non-negative sparse coding. 2002. doi: 10.1109/NNSP.2002.1030067.

Patrik O. Hoyer. Non-negative matrix factorization with sparseness constraints. *The Journal of Machine Learning Research*, 5:1457–1469, 2004.

Patrik O. Hoyer and Aapo Hyvärinen. Independent component analysis applied to feature extraction from colour and stereo images. *Network (Bristol, England)*, 11(3): 191–210, aug 2000. ISSN 0954-898X.

H Hubel and T Wiesel. Receptive fields, binocular inception and functional architecture in the cat's visual cortex. *Journal of Physiology*, 160(1):106–154, 1982. ISSN 0270-6474. doi: 10.1523/JNEUROSCI.1991-09.2009.

Aapo Hyvärinen and Erkki Oja. A Fast Fixed-Point Algorithm for Independent Component Analysis. *Neural Computation*, 9(7):1483–1492, oct 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.7.1483.

Aapo Hyvärinen and Erkki Oja. Independent component analysis: algorithms and applications. *Neural networks : the official journal of the International Neural Network Society*, 13(4-5):411–30, 2000. ISSN 0893-6080.

Nathan Intrator and Leon N. Cooper. Objective function formulation of the BCM theory of visual cortical plasticity: Statistical connections, stability conditions. *Neural Networks*, 5(1):3–17, 1992. ISSN 08936080. doi: 10.1016/S0893-6080(05)80003-6.

K Jarrett, K Kavukcuoglu, M Ranzato, and Yann LeCun. What is the best multi-stage architecture for object recognition? BT - Computer Vision, 2009 IEEE 12th International Conference on. pages 2146–2153, 2009.

M. M. Karnani, J. Jackson, I. Ayzenshtat, A. Hamzehei Sichani, K. Manoocheri, S. Kim, and R. Yuste. Opening Holes in the Blanket of Inhibition: Localized Lateral Disinhibition by VIP Interneurons. *Journal of Neuroscience*, 36(12):3471–3480, 2016. ISSN 0270-6474. doi: 10.1523/JNEUROSCI.3646-15.2016.

D. G. Kelly. Stability in contractive nonlinear neural networks. *IEEE Transactions on Biomedical Engineering*, 37(3):231–242, 1990. ISSN 00189294. doi: 10.1109/10.52325.

Arash Kermani Kolankeh, Michael Teichmann, and Fred H Hamker. Competition improves robustness against loss of information. *Frontiers in computational neuroscience*, 9(March):35, 2015. ISSN 1662-5188. doi: 10.3389/fncom.2015.00035.

Paul D King, Joel Zylberberg, and Michael R DeWeese. Inhibitory interneurons decorrelate excitatory cells to drive sparse code formation in a spiking model of V1. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 33(13):5475–85, 2013. ISSN 1529-2401. doi: 10.1523/JNEUROSCI.4188-12.2013.

Jan Johan Koenderink. The structure of locally orderless images. *International Journal of Computer Vision*, 31:159–168, 1999. ISSN 0920-5691. doi: 10.1023/A:1008065931878.

Raul Kompass. A Generalized Divergence Measure for Nonnegative Matrix Factorization. *Neural computation*, 19(3):780–791, 2007. doi: doi:10.1162/neco.2007.19.3.780.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Advances In Neural Information Processing Systems*, pages 1–9, 2012. ISSN 10495258. doi: http://dx.doi.org/10.1016/j.protcy.2014.09.007.

Sandra J Kuhlman, Nicholas D Olivas, Elaine Tring, Taruna Ikrar, Xiangmin Xu, and Joshua T Trachtenberg. A disinhibitory microcircuit initiates critical-period plasticity

in the visual cortex. *Nature*, 501(7468):543–6, 2013. ISSN 1476-4687. doi: 10.1038/nature12485.

Quoc V Le, Google Brain, and Mountain View. A Tutorial on Deep Learning Part 2: Autoencoders, Convolutional Neural Networks and Recurrent Neural Networks. pages 1–20, 2015.

Yann LeCun. Handwritten Digit Recognition with a Back-Propagation Network. *Advances in Neural Information Processing Systems*, pages 396–404, 1989. ISSN 1524-4725. doi: 10.1111/dsu.12130.

Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2323, 1998. ISSN 00189219. doi: 10.1109/5.726791.

D D Lee and H S Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–91, oct 1999. ISSN 0028-0836. doi: 10.1038/44565.

Michael Lemmon and B. V K Vijaya Kumar. Emulating the dynamics for a class of laterally inhibited neural networks. *Neural Networks*, 2(3):193–214, 1989. ISSN 08936080. doi: 10.1016/0893-6080(89)90004-X.

Ashok Litwin-Kumar, Robert Rosenbaum, and Brent Doiron. Inhibitory stabilization and visual coding in cortical circuits with multiple interneuron subtypes. *Journal of neurophysiology*, page jn.00732.2015, 2016. ISSN 1522-1598. doi: 10.1152/jn.00732.2015.

Zhi Hong Mao and Steve G. Massaquoi. Dynamics of winner-take-all competition in recurrent neural networks with lateral inhibition. *IEEE Transactions on Neural Networks*, 18(1):55–69, 2007. ISSN 10459227. doi: 10.1109/TNN.2006.883724.

Javier R Movellan. Tutorial on Gabor Filters. 49(May):1–23, 2008. ISSN 03892328. doi: 10.1016/j.neuropharm.2005.06.011.

Naila Murray, Florent Perronnin, Computer Vision Group, and Centre Europe. Generalized Max Pooling. 2014.

Jim Mutch and David G. Lowe. Multiclass Object Recognition Using Sparse, Localized Features. *Computer Vision and Pattern Recognition, IEEE Conference on*, pages 11–18, 2006. ISSN 10636919. doi: 10.1109/CVPR.2006.200.

Mohammad Norouzi, Mani Ranjbar, and Greg Mori. Stacks of convolutional restricted boltzmann machines for shift-invariant feature learning. *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2009*, 5(2):2735–2742, 2009. ISSN 1063-6919. doi: 10.1109/CVPRW.2009.5206577.

Erkki Oja. A simplified neuron model as a principal component analyzer. *Journal of mathematical biology*, 15(3):267–273, 1982. ISSN 0303-6812. doi: 10.1007/BF00275687.

Bruno A. Olshausen and David J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996. ISSN 0028-0836. doi: 10.1038/381607a0.

Randall C. O'Reilly, Dean Wyatte, Seth Herd, Brian Mingus, and David J. Jilk. Recurrent processing during object recognition. *Frontiers in Psychology*, 4(APR):1–14, 2013. ISSN 16641078. doi: 10.3389/fpsyg.2013.00124.

Claudio A. Perez, Cristian A. Salinas, Pablo A. Est??vez, and Patricia M. Valenzuela. Genetic design of biologically inspired receptive fields for neural pattern recognition. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 33(2):258–270, 2003. ISSN 10834419. doi: 10.1109/TSMCB.2003.810441.

Son Lam Phung and Abdesselam Bouzerdoum. A Pyramidal Neural Network For Visual Pattern Recognition. 18(2):329–343, 2007.

Nicolas Pinto, David D. Cox, and James J. DiCarlo. Why is real-world visual object recognition hard? *PLoS Computational Biology*, 4(1):0151–0156, 2008. ISSN 1553734X. doi: 10.1371/journal.pcbi.0040027.

Marc'aurelio Ranzato, Y. Lan Boureau, and Yann L. Cun. Sparse Feature Learning for Deep Belief Networks. *Advances in Neural Information Processing Systems*, (Mcmc):1185–1192, 2008.

Martin Rehn and Friedrich T Sommer. A network that uses few active neurones to code visual input predicts the diverse shapes of cortical receptive fields. pages 135–146, 2007. doi: 10.1007/s10827-006-0003-9.

Maximilian Riesenhuber and Tomaso Poggio. Hierarchical models of object recognition in cortex. *Nature neuroscience*, 2(11):1019–25, 1999. ISSN 1097-6256. doi: 10.1038/14819.

Dario L Ringach. Spatial structure and symmetry of simple-cell receptive fields in macaque primary visual cortex. *Journal of neurophysiology*, 88(1):455–63, jul 2002. ISSN 0022-3077.

Raul Rojas. Introduction. *Neural Networks, A systematic Introduction*, page 257, 1996.

Christopher J Rozell, Don H Johnson, Richard G Baraniuk, and Bruno A Olshausen. Sparse coding via thresholding and local competition in neural circuits. *Neural Computation*, 20(10):2526–63, 2008. ISSN 08997667. doi: 10.1162/neco.2008.03-07-486.

Bernardo Rudy, Gordon Fishell, SooHyun Lee, and Jens Hjerling-Leffler. Three groups of interneurons account for nearly 100% of neocortical GABAergic neurons. *Developmental Neurobiology*, 71(1):45–61, 2011. ISSN 19328451. doi: 10.1002/dneu.20853.

David Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986. ISSN 0028-0836. doi: 10.1038/323533a0.

David Rumelhart, David E. Zipser. Feature discovery by competitive learning., 1985.

Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted Boltzmann Machines for Collaborative Filtering. *Proceedings of the 24th ACM International Conference on Machine Learning (ICML)*, pages 791–798, 2007. doi: 10.1145/1273496.1273596.

Terence D. Sanger. Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural Networks*, 2(6):459–473, 1989. ISSN 08936080. doi: 10.1016/0893-6080(89)90044-0.

Thomas Serre and Tomaso Poggio. Object Recognition with Features Inspired by Visual Cortex. ISSN 1063-6919. doi: 10.1109/CVPR.2005.254.

Michael N. Shadlen and W. T. Newsome. The variable discharge of cortical neurons: implications for connectivity, computation, and information coding. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 18(10):3870–96, 1998. ISSN 0270-6474. doi: 0270-6474/98/183870-27$05.00/0.

Michael W Spratling. Learning image components for object recognition. *The Journal of Machine Learning Research*, 7(5):793–815, 2006. ISSN 1532-4435.

Michael W Spratling. Predictive coding as a model of response properties in cortical area V1. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 30(9):3531–43, mar 2010. ISSN 1529-2401. doi: 10.1523/JNEUROSCI.4911-09.2010.

Michael W Spratling. Unsupervised learning of generative and discriminative weights encoding elementary image components in a predictive coding model of cortical function. *Neural Computation*, 24(1):60–103, 2012.

Michael W Spratling. A review of predictive coding algorithms. *Brain and Cognition*, pages 1–8, 2016. ISSN 02782626. doi: 10.1016/j.bandc.2015.11.003.

Michael W Spratling, K De Meyer, and R Kompass. Unsupervised learning of overlapping image components using divisive input modulation. *Computational intelligence and neuroscience*, 2009(381457):381457, jan 2009. ISSN 1687-5273. doi: 10.1155/2009/381457.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: prevent NN from overfitting. 15:1929–1958, 2014.

Carlos Stein and Naves De Brito. Nonlinear Hebbian learning as a universal principle in unsupervised feature learning. pages 1–30, 2015.

Michael Teichmann, Jan Wiltschut, and Fred H. Hamker. Learning invariance from natural images inspired by observations in the primary visual cortex. *Neural computation*, 24(5):1271–96, may 2012. ISSN 1530-888X. doi: 10.1162/NECO_a_00268.

Fok Hing Chi Tivive and Abdesselam Bouzerdoum. Efficient training algorithms for a class of shunting inhibitory convolutional neural networks. *IEEE Transactions on Neural Networks*, 16(3):541–556, 2005. ISSN 10459227. doi: 10.1109/TNN.2005.845144.

Original Contrib Ution. A Cortical Model of Winner-Take-All Competition Via Lateral Inhibition. 5:47–54, 1992.

J H van Hateren and A van der Schaaf. Independent component filters of natural images compared with simple cells in primary visual cortex. *Proceedings. Biological sciences / The Royal Society*, 265(1394):359–66, mar 1998. ISSN 0962-8452. doi: 10.1098/rspb.1998.0303.

Daniëlle van Versendaal and Christiaan N Levelt. Inhibitory interneurons in visual cortical plasticity. *Cellular and molecular life sciences : CMLS*, pages 3677–3691, 2016. ISSN 1420-682X. doi: 10.1007/s00018-016-2264-4.

Christoph von der Malsburg. Self-organization of orientation sensitive cells in the striate cortex. *Kybernetik*, 14(2):85–100, 1973. ISSN 0340-1200. doi: 10.1007/BF00288907.

Li Wan, Matthew Zeiler, Sixin Zhang, Yann LeCun, and Rob Fergus. Regularization of neural networks using dropconnect. *Icml*, (1):109–111, 2013.

Michael S. Wehr and Anthony M. Zador. Balanced inhibition underlies tuning and sharpens spike timing in auditory cortex. *Nature*, 426(6965):442–6, 2003. ISSN 1476-4687. doi: 10.1038/nature02116.

Jan Wiltschut and Fred H. Hamker. Efficient coding correlates with spatial frequency tuning in a model of V1 receptive field organization. *Visual neuroscience*, 26(1):21–34, 2009. ISSN 1469-8714. doi: 10.1017/S0952523808080966.

Liping Xiao and Jiqian Liu. Lateral Inhibition Underlying Suppression of Neuronal Activity and Sparse Coding. 3(2):144–148, 2015. doi: 10.18178/joig.3.2.144-148.

Matthew D. Zeiler and Rob Fergus. Stochastic Pooling for Regularization of Deep Convolutional Neural Networks. *International Conference on Representation Learning*, pages 1–9, 2013.

Matthew D. Zeiler and Rob Fergus. Visualizing and Understanding Convolutional Networks arXiv:1311.2901v3 [cs.CV] 28 Nov 2013. *Computer Vision–ECCV 2014*, 8689:818–833, 2014. ISSN 978-3-319-10589-5. doi: 10.1007/978-3-319-10590-1_53.

Joel Zylberberg, Jason Timothy Murphy, and Michael Robert DeWeese. A Sparse Coding Model with Synaptically Local Plasticity and Spiking Neurons Can Account for the Diverse Shapes of V1 Simple Cell Receptive Fields. *PLoS Computational Biology*, 7:e1002250, oct 2011. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1002250.

Appendix

1 Gabor Filter

Gabor filters belong to the family of band-pass filters, meaning that they detect a specific frequency range in a signal and omit the other frequencies. In addition, Gabor filters are capable of detecting a frequency range of a signal within a limited spatial window and a specific direction (Movellan [2008]). These filters are widely used for edge detection in images. Developing tools for designing Gabor filters is an essential part of many image processing tasks. In this work, we have used a software developed in the AI Department at TU-Chemnitz and written in Matlab for designing Gabor filters as well as for teaching purposes. Two main problems have been solved in this work: finding a relation between the frequency and width (sigma) of the Gabor filter and removing the DC component of the filter. As the filters where designed to be used for image processing, here we concentrate on 2 dimensional filters. In all experiments, 11*11 pixel windows were used for the filters. Gabor Filter

A Gabor filter is made of a Gaussian signal multiplied with a cosine signal Dayan and Abbott [2001]:

$$Gabor(x', y') = Gauss(x', y') \cos(2\pi f_0 x' + \varphi)$$

where:

$$Gauss(x) = K\frac{1}{\sqrt{2\pi\sigma_x}}e^{(\frac{-x^2}{\sigma_x^2})}\frac{1}{\sqrt{2\pi\sigma_y}}e^{(\frac{-y^2}{\sigma_y^2})}$$

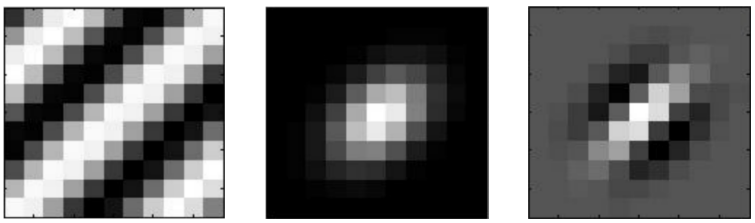
f_0 is the frequency and φ is the phase of the cosine. The rotation of the filter is described as:

$$x' = x\cos\theta + y\sin\theta$$

and

$$y' = -x\sin\theta + y\cos\theta$$

By changing θ we can change the direction in which we want to detect the edges. On Figure 1 you can see the sinusoidal and Gaussian components of a filter as well as the filter itself.



(a) (b) (c)

Figure 1: (a) a two dimensional sinusoidal signal rotated by 45 degrees. (b) two dimensional Gaussian signal rotated by 45 degrees. (c) the Gabor filter produced by multiplying the sinusoidal and Gaussian components in (a) and (b).

1.1 Filtering

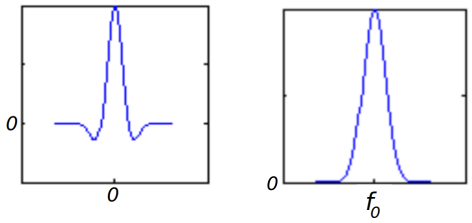
The filtering process in fact is convolving a filter with the target signal in the spatial domain. This was done using the conv2 function in Matlab. The convolution in the spatial domain can be interpreted as multiplication in the frequency domain.

$$r = image * filter$$

$$F(r) = F(image).F(filter)$$

where 'r' is the result of filtering, * denotes convolution and F() indicates the frequency domain.

The Fourier transform of the Gator filter is a Gaussian signal centered at the central frequency of the filter. This is shown in one-dimensional form on Figure 2 .The frequency of the sinusoidal signal, f_0, is the central frequency of the filter.



(a) (b)

Figure 2: (a) One-dimensional Gabor filter. (b) The Fourier transform of the Gabor filter with the central frequency f_0.

As the result of the multiplication in the frequency domain, the amplitudes of the frequencies of the target signal closer to the central frequency of the filter are strengthened and the others are weakened. The role of the Gaussian component is strengthening the frequencies closer to the frequency of the sinusoidal signal.

1.2 Filter design

The width and height of the filter are defined by the width and hight of the Gaussian component. Defining the correct width and height for the Gabor is essential in designing good filters. We experimentally found that a good filter capable of detecting the narrowest edges with a specific frequency should have its borders on zero and have two negative and one positive peaks Figure 3 . We also found that choosing

$$\sigma_x = \frac{1}{\sqrt{(2\pi f_0)}}$$

and

$$\sigma_y = \frac{3\sigma_x}{2}$$

guarantees having filters with the mentioned specifications.

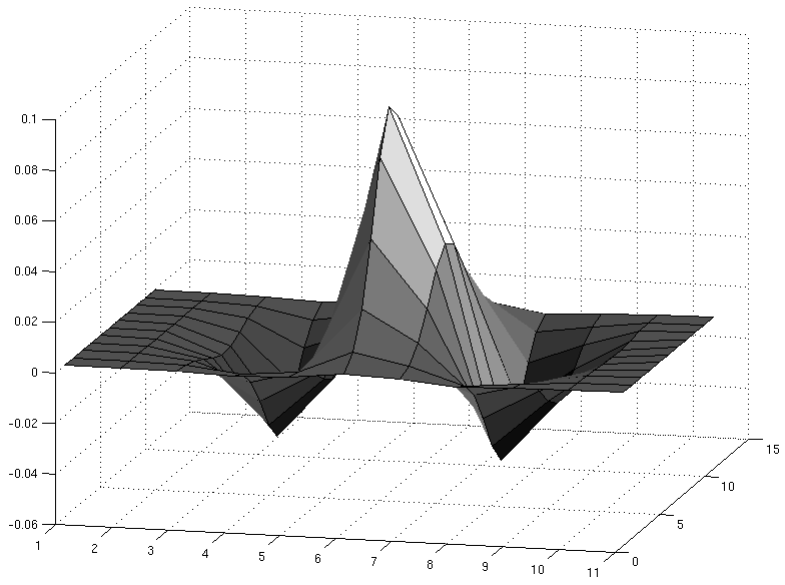


Figure 3: A two-dimensional Gabor filter with its borders located on zero and with two negative and one positive peaks.

The DC component

In the ideal case, the Gabor filter should have zero mean. It means that if the filter is convolved with a flat signal or a signal with a frequency much higher or much lower than the central frequency of the filter, the result should be a zero signal. This is due to the fact that the positive and negative parts of the signal cancel each other in the integral (convolution). Although, in fact we always have a mean bigger or lower than zero (Movellan J. 2008). The mean of the filter is also called the DC of the filter. Convolving

an image with a filter containing non-zero DC results to an output contacting the whole image in addition to the highlighted edges (Figure 4).

(a) (b)

Figure 4: (a)The original image (b)The filtered image with a none-zero DC Gabor filter

A common way of removing the DC component from a filter is subtracting the mean (DC) of the filter from it. Although this method will remove the DC component, it deforms the filter by shifting its borders from the zero (Figure 5).This will add an artificial edge to the borders of the filter.

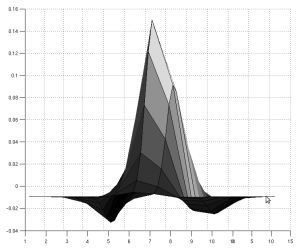


Figure 5: The effect of subtracting the mean of the filter from it to remove the DC. The filters borders are not located on zero.

In our method, we have subtracted a Gaussian signal from the Gabor filter which results in saving the filters shape and having its borders on zero. The problem is finding the

appropriate Gaussian signal to be subtracted from the Gabor filter. In order to have the right Gaussian signal, we divided a Gaussian signal, the same as the Gaussian component of the filter, by its mean to have a signal with the mean 1 and then multiplied the result with the DC of the filter. The result is a Gaussian signal with the mean equal to the DC of the filter. If we subtract this signal from the filter, in fact we have subtracted the DC from its the mean of the filter and reduced the DC (mean) to zero. On figure 6 you can see an image with a filter without the DC component.

The idea is that if we divide a curve by its mean, the mean of the resulting curve will be one:

$$g = Gauss(x, y) = (g_1, g_2, ..., g_n)$$

$$M = mean(g_1, g_2, ..., g_n) = \frac{\Sigma g_i}{n}$$

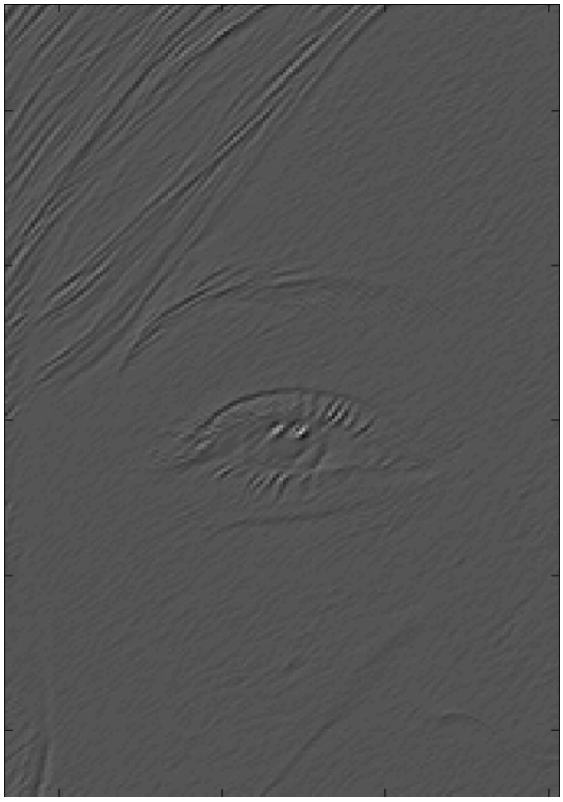
$$K = mean(\frac{g_1}{M}, \frac{g_2}{M}, ...\frac{g_n}{M}) = \frac{\frac{\Sigma g}{n}}{M} = \frac{M}{M} = 1;$$

and by multiplying it to DC or the mean of the Gabor filter, the mean of this Gaussian signal will be changed to the DC of the Gaussian. By subtracting it from the Gabor filter, we will have a filter with the mean value or DC of zero (Figure 3). On Figure 6 you can see the result of filtering with a zero-DC filter.

$$Gabor'(x', y') = Gabor(x', y') - \frac{Gauss(x', y')}{mean(Gauss)}.mean(Gabor)$$

$$(mean(Gabor) = DC)$$

(a)



(b)

Figure 6: (a)The original image (b)The filtered image with a zero DC Gabor filter

1.3 Complex set of Gabor Filters

Complex V1 cell are capable of detecting edges regardless of their phase. In this study we have simulated the phase invariance by calculating the energy (Hoyer [2002]) of the results of filtering an image with a filter which was 4 time by 0, 45, 90 and 135 degrees shifted: if we redefine the Gabor filter as follows:

$$Gabor(x', y', \varphi) = Gauss(x', y') \cos(2\pi f_0 x' + \varphi)$$

then the energy of the filtered images will be:

$$Energy(x', y') = \sqrt{I_{f0}^2, I_{f1}^2, I_{f2}^2, I_{f3}^2}$$

$$I_{f0} = (Gabor(x', y', 0)) * Image$$

$$I_{f1} = (Gabor(x', y', \frac{\pi}{4})) * Image$$

$$I_{f2} = (Gabor(x', y', \frac{2\pi}{4})) * Image$$

$$I_{f3} = (Gabor(x', y', \frac{3\pi}{4})) * Image$$

As in the case any of the 4 filters detects an edge the Energy will have a high value, we have achieved phase invariance in edge detecting. The images we used as the input to our network were filtered with the Energy model including filters with following parameters:

$$f_0 = 0.20691;$$

$$\varphi = 0, 45, 90, 135;$$

$$\sigma_x = 0.87704;$$

$$\sigma_y = 1.3156;$$